INDUSTRIAL AUTOMATION
MARCH 2025
ISSUE 416

circuitcellar.com

circuit cellar

Inspiring the Evolution of Embedded Design

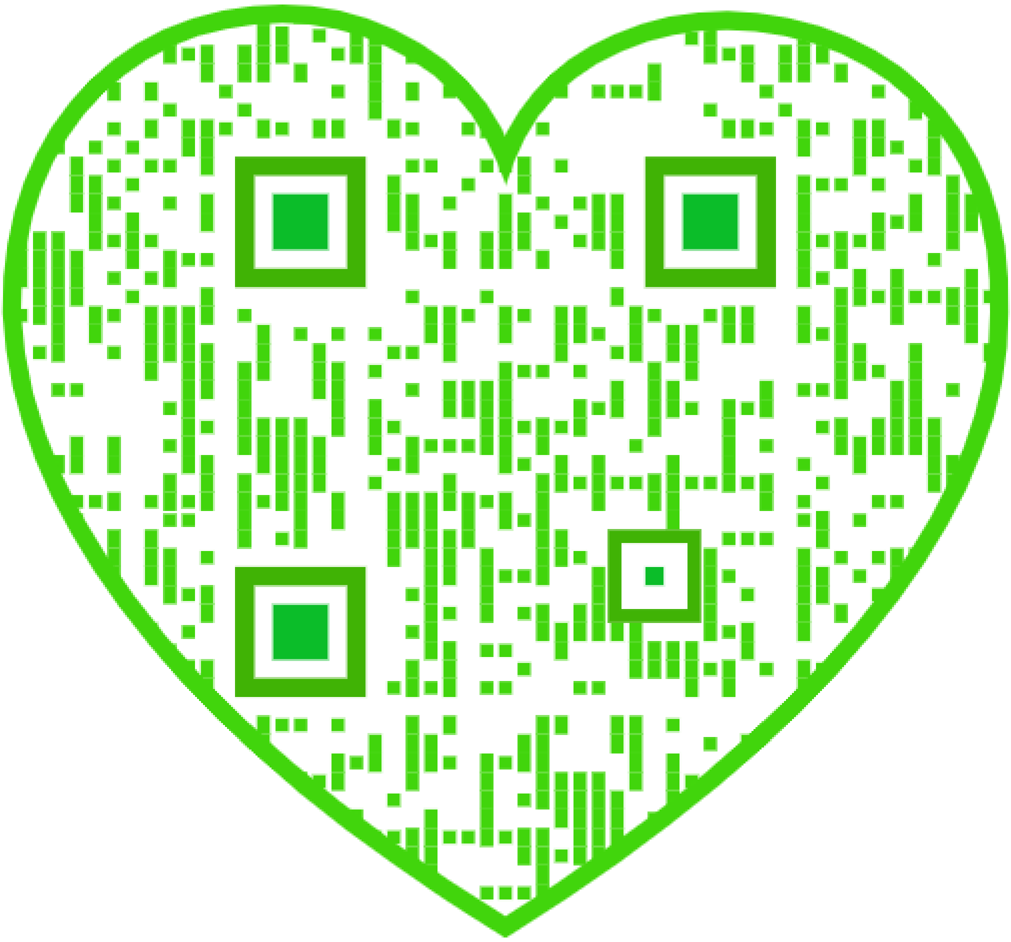
INDUSTRIAL AUTOMATION

DSPs and DSCs | Accelerating AI — Handwriting Recognition | Kinematics in µLisp | Learning the Intricacies of Bluetooth LE Audio | AI-Driven Smart Home Security System via Raspberry Pi

Quality Monitor with a Blues Satellite Link Module | Exploring the RP2350 Security

The Future of Voice-Enabled AI

$9.00US $10.00CAN



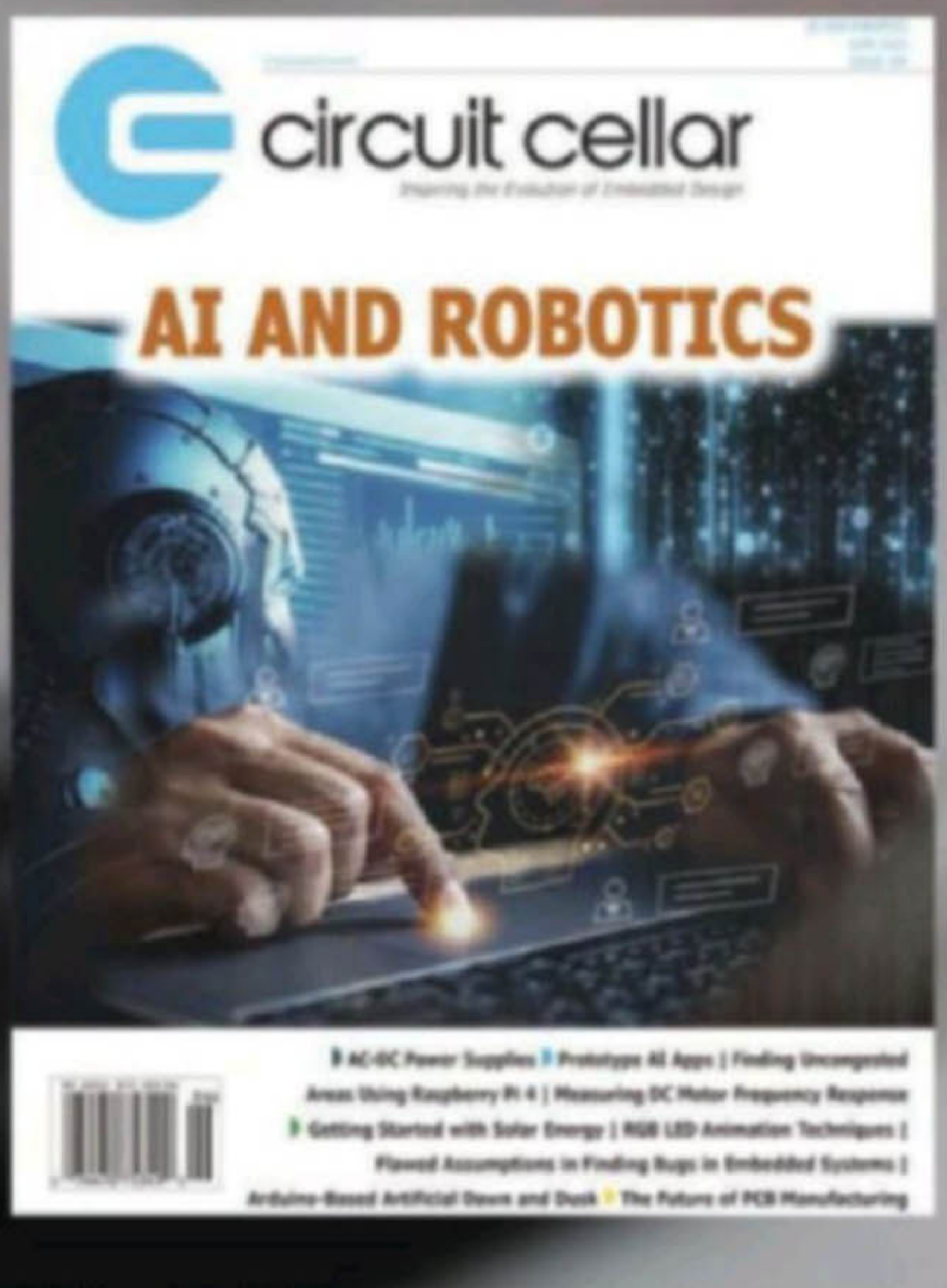
Imagine what you will
Discover
circuit cellar
VAULT
CC VAULT
circuit cellar
SMART BUILDINGS AND CITIES
circuit cellar
DATA ACQUISITION TECHNOLOGY
circuit cellar
PCB DESIGN & PROTOTYPING
circuit cellar
TINY EMBEDDED BOARDS

400+ ISSUES

Get the complete *Circuit Cellar* issue archive and article code stored on a stylish, durable and portable USB flash drive. You can easily keep your CC Vault archive up to date by purchasing subsequent issues from our webshop or by downloading issues with a *Circuit Cellar* Digital Subscription. Issues appear in searchable PDF format.

cc-webshop.com

circuit cellar
WEARABLE DEVICES


circuit cellar
AI AND ROBOTICS

circuit cellar
DRONE DEVELOPMENT AND USAGE

circuit cellar
MCU DEVELOPMENT TOOLS

Complete archive includes PDFs of all issues in print through date of purchase.

INPUT Voltage

Why Save Time?

In today's fast-paced world, time is a precious commodity. We often hear the question, "What time do you have?" But when we shift our perspective and think of time as a resource, the question becomes, "How much time do you have?" Automation is designed to free up time and resources. Taking the burden of the mundane and repetitive, giving back the hours. Increasing the productivity in manufacturing and day to day life. The future of technology is likely to be dominated by innovations that continue to improve efficiency and automate tasks, further transforming how we work and live. Take AI, the most talked about recent technological development. AI is being incorporated into manufacturing at a breathtaking pace, automating tasks such as quality control and predictive maintenance, freeing up human workers to focus on more complex and creative tasks.

In an article co-authored by Jichao Yang and Yiyang Zhao, they discuss another use for AI as they outline a new process for digit recognition. Enhanced by FPGA-powered Convolutional Neural Networks (CNNs), which advances deep learning for computer vision to recognize handwritten numerical digits. Their design achieved 91% accuracy, providing valuable insights into using FPGAs to improve the speed and precision of digit recognition.

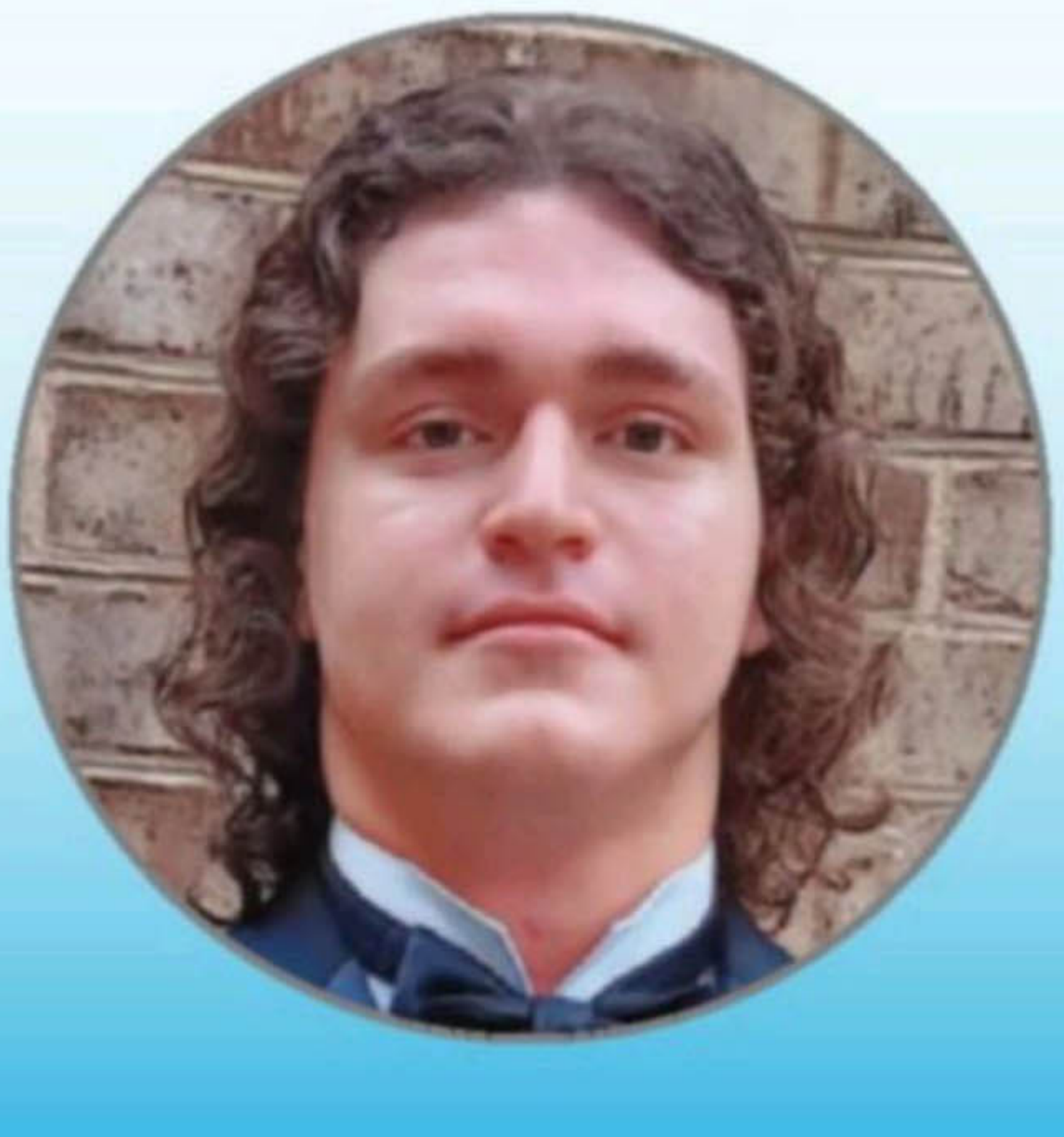
In the first part of a new series, Dhairya Parikh explores an innovative application of AI when he creates a smart home security system using a Raspberry PI 5 and attached camera modules. This project leverages an AI kit that includes a dedicated GPU for AI-related tasks, alleviating some of the challenges of running AI applications on the Raspberry Pi, which is traditionally limited by its low-power CPU and GPU combination. This introductory article provides a comprehensive overview of the hardware setup and AI model installation. Dive into this fun and practical project to enhance your understanding of both Raspberry Pi configurations and the potential of open-source AI technology!

Continuing the trend of AI, this month’s Datasheet feature covers digital signal processors and how AI, among other things is being used to improve signal clarity and processing capabilities among audio, automotive, and healthcare purposes. Compare the latest offerings from Analog Devices, AKM, Aquila, Microchip, MaxLinear, NXP, Texas Instruments, and XMOS.

Nick Hunn's article presents an in-depth review of the new Bluetooth Low Energy (LE) Audio specifications, highlighting their potential to revolutionize audio technology. The article explains the new audio middleware, Broadcast Isochronous Streams (BISes), and the Generic Audio Framework (GAF). Bluetooth LE Audio introduces improvements such as bidirectional audio, broadcast capabilities for shared listening experiences, and the high-efficiency LC3 codec. Hunn's expertise helps developers understand complicated features and user-end benefits, including new volume control specifications

Raul Alvarez-Torrico's article, the third in a series, delves into using µLisp to control MCUs, demonstrating the creation and use of lists and arrays and the kinematics required to control a robotic arm. The article explains that µLisp is a dialect of the Lisp programming language tailored for MCUs with limited RAM. Demonstrating the implementation of forward kinematics for a mini robotic arm using homogeneous transformations (HT) and introduces a matrix multiplication extension to µLisp.

If you have a project that you’re working on that might be of interest to our readers, contact me. It could be featured in a future issue of *Circuit Cellar*, possibly helping others along their own journey.



Caleb Smith

Editor-In-Chief
csmith@circuitcellar.com

Issue 416 March 2025 | ISSN 1528-0608

CIRCUIT CELLAR® (ISSN 1528-0608) is published monthly by:

KCK Media Corp.
PO Box 417, Chase City, VA 23924

Periodical rates paid at Chase City, VA, and additional offices. One-year (12 issues) subscription rate US and possessions $50, Canada $65, Foreign/ ROW $75. All subscription orders payable in US funds only via Visa, MasterCard, international postal money order, or check drawn on US bank.

SUBSCRIPTION MANAGEMENT

Online Account Management: circuitcellar.com/account
Renew | Change Address/E-mail | Check Status

CUSTOMER SERVICE

E-mail: customerservice@circuitcellar.com

Phone: 434.533.0246

Mail: Circuit Cellar, PO Box 417, Chase City, VA 23924

Postmaster: Send address changes to
Circuit Cellar, PO Box 417, Chase City, VA 23924

NEW SUBSCRIPTIONS

circuitcellar.com/subscription

ADVERTISING

Contact: Hugh Heinsohn

Phone: 757-525-3677

Fax: 888-980-1303

E-mail: hheinsohn@circuitcellar.com
Advertising rates and terms available on request.

NEW PRODUCTS

E-mail: product-editor@circuitcellar.com

HEAD OFFICE

KCK Media Corp.
PO Box 417
Chase City, VA 23924
Phone: 434-533-0246

COPYRIGHT NOTICE

Entire contents copyright © 2025 by KCK Media Corp. All rights reserved. Circuit Cellar is a registered trademark of KCK Media Corp. Reproduction of this publication in whole or in part without written consent from KCK Media Corp. is prohibited.

DISCLAIMER

KCK Media Corp. makes no warranties and assumes no responsibility or liability of any kind for errors in these programs or schematics or for the consequences of any such errors printed in Circuit Cellar®. Furthermore, because of possible variation in the quality and condition of materials and workmanship of reader-assembled projects, KCK Media Corp. disclaims any responsibility for the safe and proper function of reader-assembled projects based upon or from plans, descriptions, or information published in Circuit Cellar®.

The information provided in Circuit Cellar® by KCK Media Corp. is for educational purposes. KCK Media Corp. makes no claims or warrants that readers have a right to build things based upon these ideas under patent or other relevant intellectual property law in their jurisdiction, or that readers have a right to construct or operate any of the devices described herein under the relevant patent or other intellectual property law of the reader’s jurisdiction. The reader assumes any risk of infringement liability for constructing or operating such devices.

© KCK Media Corp. 2025 Printed in the United States

OUR NETWORK

circuit cellar
LinuxGizmos
audioxpress
VOICE COIL
LIS LOUDSPEAKER INDUSTRY SOURCEBOOK

SUPPORTING COMPANIES

Blackmagic Design	9
Beningo Embedded Group	51
BusBoard Prototype Systems	15
CCS, Inc.	61
EmbeddedTS	61, C4
Embedded World 2025	55
Myir	35
Recom	60
SeaFire Micros, Inc.	C3

NOT A SUPPORTING COMPANY YET?
Contact Hugh Heinsohn
hugh@circuitcellar.com,
Phone: 757-525-3677,
Fax: 888-980-1303
to reserve space in the
next issue of *Circuit Cellar.*

THE TEAM

FOUNDER Steve Ciarcia
PUBLISHER KC Prescott
CONTROLLER Chuck Fellows
EDITOR-IN-CHIEF Caleb Smith
CONTRIBUTING EDITOR Brian Millier
CONTENT MANAGER- EU Jurij Mikeln
SENIOR ASSOCIATE EDITOR Shannon Becker
TECHNICAL COPY EDITOR Carol Bower
PROJECT EDITORS Ken Davidson
David Tweed
MARKETING MANAGER Tori Zienka
ADVERTISING COORDINATOR
Diana Winsor
ADVERTISING SALES REP.
Hugh Heinsohn

COLUMNISTS
Stuart Ball (Start to Finish)
Jacob Beningo (Modern Embedded Insights)
Bob Japenga (Embedded in Thin Slices)
Brian Millier (Picking Up Mixed Signals)
Colin O'Flynn (Embedded Systems Essentials)

FEATURES

4 **Accelerating AI**
Handwritten Digit Recognition with FPGA-Powered CNNs
By Jichao Yang and Yiyang Zhao

10 **Learn Lisp Programming Using MCUs**
Part 3—Forward Kinematics in μLisp
By Raul Alvarez-Torrico

18 **Introducing Bluetooth LE Audio**
Part 2—Learning the Intricacies of these New Specifications
By Nick Hunn

24 **Create an AI-Driven Smart Home Security System**
Part 1— Setup and First Steps of the Raspberry Pi AI Kit
By Dhairya Parikh

---

30 TECHNOLOGY FEATURE
Industrial Automation
Revolutionizing Production with Innovation
By Caleb Smith

36 DATASHEET
DSPs and DSCs
Smarter Algorithms and Beyond
By Caleb Smith

COLUMNS

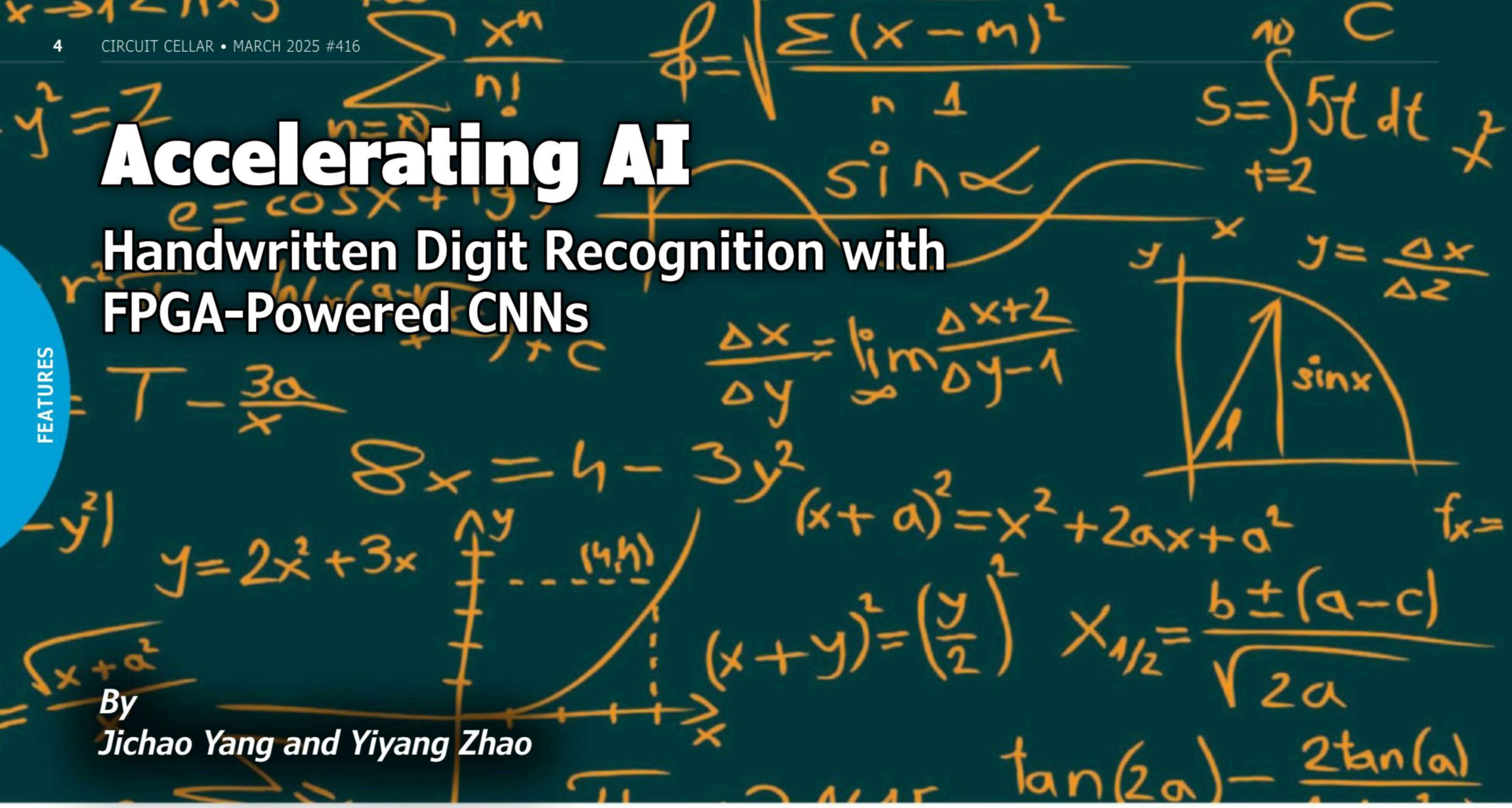
40 Picking Up Mixed Signals
An Air Quality Monitor with a Blues Satellite Link Module
By Brian Millier

52 Embedded System Essentials
Exploring the RP2350 Security
Raspberry Pi's Novel "Security through Transparency" Approach
By Colin O'Flynn

62 TECH THE FUTURE
The Future of Voice-Enabled AI
Industry Advancements in MEMS Microphones for Voice AI
Acoustic Activity Detection and High Acoustic Overload Point Transform Voice Interaction
By Uday Mudoi

57 : PRODUCT NEWS
60 : TEST YOUR EQ

@editor_cc
@circuitcellar
circuitcellar
CircuitCellar



Accelerating AI

Handwritten Digit Recognition with FPGA-Powered CNNs

By
Jichao Yang and Yiyang Zhao

Discover how FPGA-powered Convolutional Neural Networks (CNNs) revolutionize handwritten digit recognition, achieving impressive speeds and accuracy. This article delves into the innovative implementation on the DE1-SoC platform, showcasing the seamless integration of hardware and advanced machine learning techniques.

In the fast-paced world of artificial intelligence, deep learning—a key branch of machine learning—has gained significant attention, especially in fields such as computer vision and image processing. Among the various deep learning models, Convolutional Neural Networks (CNNs) have become the go-to choice for image recognition and analysis.

Since Yann LeCun introduced the LeNet-5 model in 1998 for handwritten digit recognition, CNNs have been widely adopted. However, running these models in programming languages such as C or Python can be slow due to their sequential processing nature and memory bottlenecks. To address those issues, we implemented the CNN model for digit recognition on an FPGA with Verilog, a hardware description language that allows for hardware-level parallelism and efficient data management through on-board memory (SRAM). The implementation utilizes parallelization for both computations and memory accesses, enabling faster processing.

This article provides an overview of how we implemented a CNN model on the DE1-SoC, a hardware design platform that includes the Altera Cyclone V FPGA and dual-core Cortex-A9 embedded cores, utilizing hardware-level parallelism. It explains the CNN architecture, the DE1-SoC board we used, the communication between the FPGA and hard processor system (HPS), and outlines the digit recognition process.

TRADITIONAL APPROACH FOR HANDWRITTEN DIGIT RECOGNITION

Earlier digit recognition algorithms include k-Nearest Neighbors (k-NN), Support Vector Machines (SVM), and basic Artificial Neural Networks (ANN). While k-NN is simple, it struggles with high-dimensional data and is computationally expensive. SVMs are effective for small datasets but require extensive tuning for larger, more complex data. Basic ANNs lack the capacity to automatically extract features, making them less accurate. In contrast, CNNs are designed specifically for image data. They automatically learn hierarchical features, which makes them significantly more accurate and efficient for tasks such as handwritten digit recognition.

In this article, we explore the representation of digit images in grayscale, where each pixel corresponds to an element in a matrix. The grayscale values are 8-bit, ranging from 0 (black) to 255 (white). Traditionally, recognizing digit images involves analyzing pixel-level features such as intensity, shape, and texture. A threshold is often applied to determine whether specific features exceed certain limits. This conventional approach operates without a loss function, which is used to guide the optimization process to improve accuracy and does not require labeled data, making it an unsupervised learning technique.

With the rapid advancements in machine learning, there is growing interest in applying these techniques to the problem of handwritten digit recognition, offering potential improvements over traditional methods. One of the most influential techniques is CNN.

CNN OVERVIEW

First, let's have a look at how CNN works. It takes an image as input and produces a vector output representing the predicted class. By progressively transforming the input image through multiple layers, each specializing in different feature extraction tasks, CNNs can recognize complex patterns.

As shown in **Figure 1**, a good starting point to introduce CNN is to list out the functional components: convolution layer, Rectified Linear Unit (ReLu) module, max-pooling layer and fully connected layer.

The convolution layer can be thought of as a feature detector. It performs feature detection by applying a filter (a small matrix of weights) over each section of the image, calculating the weighted sum, and producing a new value that highlights specific patterns, such as edges or textures, in that area. By sliding this filter across the entire image, it helps the model understand the unique features that make up different objects or digits. In our case, there are three 3×3 pixel filters sliding across the 28×28 pixel image, returning three 26×26 pixel images. The selection of image and filter sizes is influenced by the limited memory capacity of our board, which we'll explore further in the next section.

After the image is filtered, the ReLU module helps the model focus on the important parts by "turning off" negative values, essentially removing irrelevant data and making the positive information stand out for further analysis. After applying the ReLu function, the positive values remain and negative values have been set to zero.

Imagine zooming out from an image and only keeping the most important parts. That's what the max-pooling layer does. It reduces the size of the data by keeping only the most significant information from each section, which makes the network faster and more efficient without losing too much detail. Mathematically speaking, the max-pooling layer reduces the spatial dimensions by selecting the maximum value from each subregion of the feature map produced by the convolutional layer.

In the final step, the fully connected layer takes everything the previous layers have learned and makes the decision. It connects all the dots (or pixels, in this case) and outputs the final result, like recognizing the digit in a handwritten number. It's the part that delivers the answer based on all the processed data. The output matrix size from the max-pooling layer is 13×13×3, containing a total of 507 values after flattening. The fully connected layer can be mathematically represented as Equation 1:

$$\begin{bmatrix} fc_0 \\ \cdots \\ fc_9 \end{bmatrix} = \begin{pmatrix} a_{0,0} & \cdots & a_{0,506} \\ \vdots & \ddots & \vdots \\ a_{10,0} & \cdots & a_{10,506} \end{pmatrix} \begin{bmatrix} mp_0 \\ \cdots \\ mp_{506} \end{bmatrix}$$

where: "fc" is the output of the fully connected layer, the number with the highest index will be picked as the recognition result. "A" is the weight matrix of the fully connected layer and "mp" the output from the max-pooling layer. The final recognition result is determined by selecting the index of the highest value in the "fc" output, which corresponds to the predicted digit.

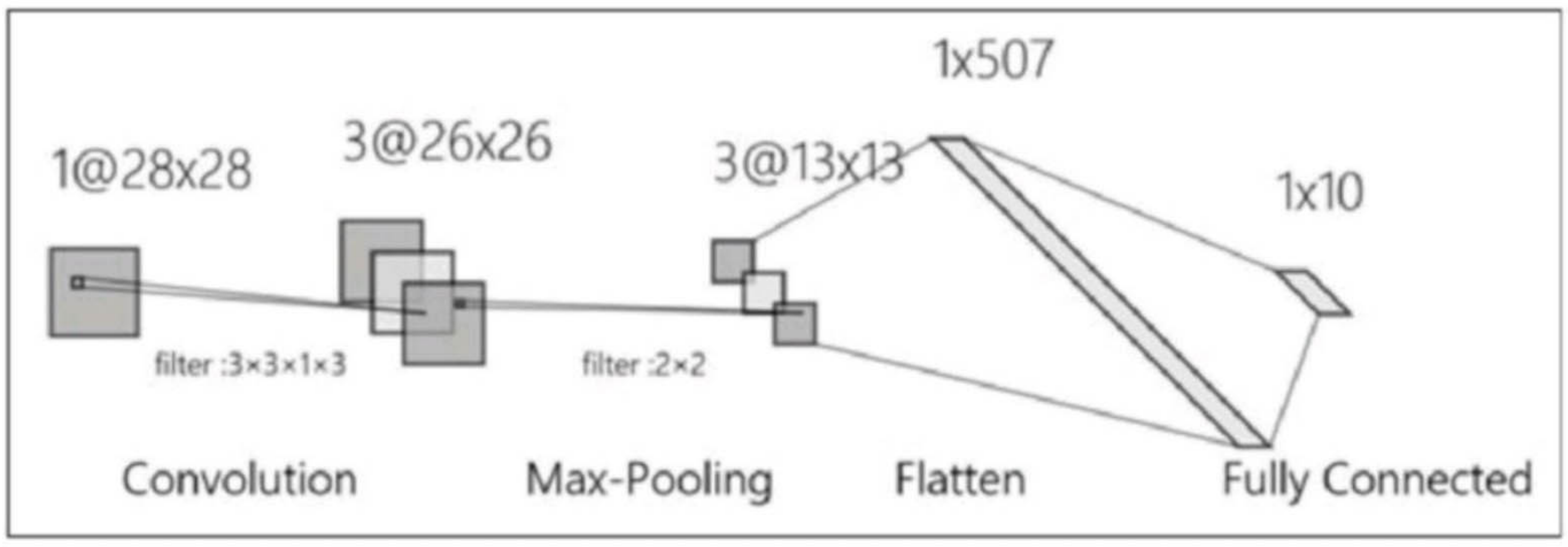

1x507
1@28x28
3@26x26
3@13x13
1x10
filter :3×3×1×3
filter :2×2
Convolution
Max-Pooling
Flatten
Fully Connected

FIGURE 1
CNN structure of this project.

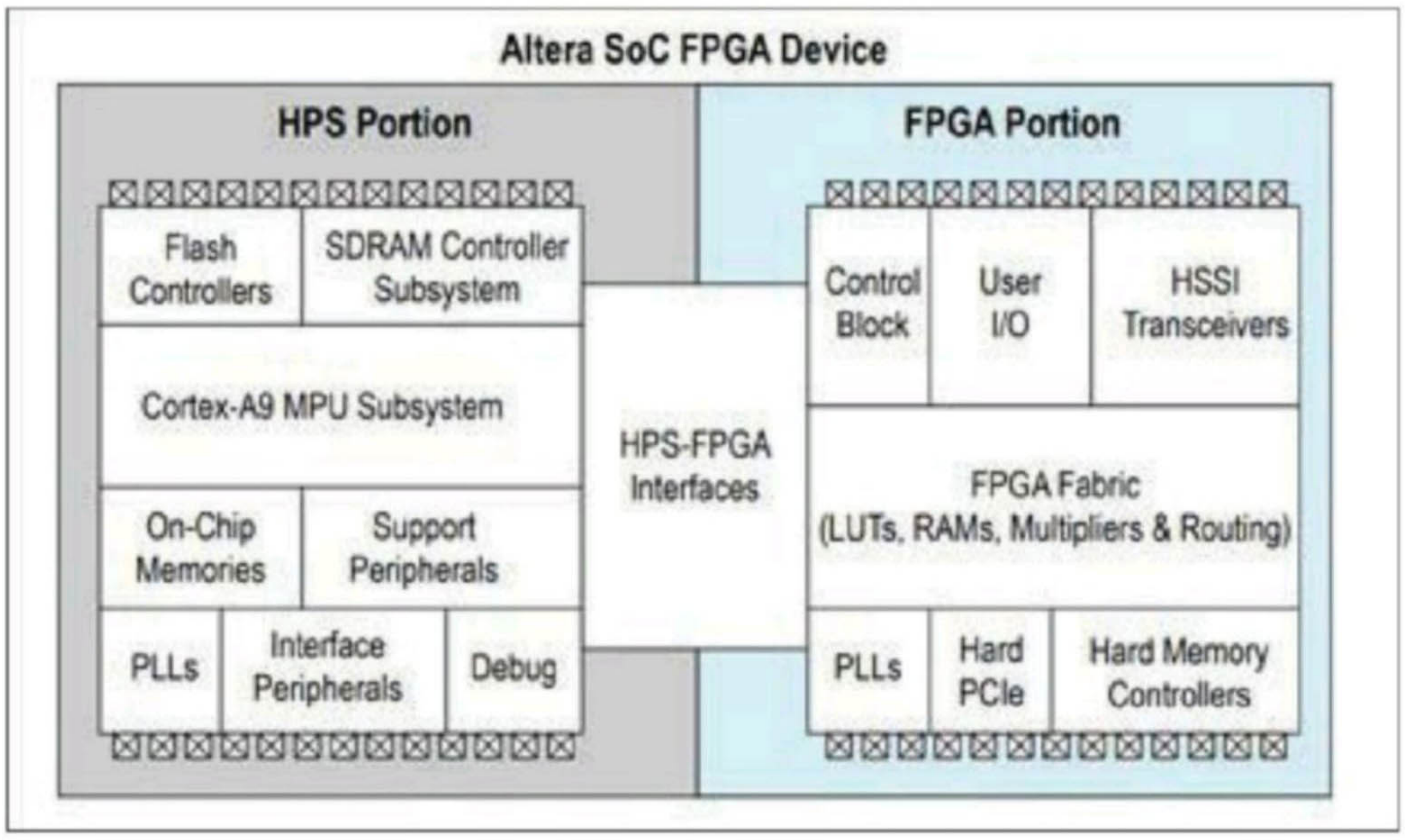

Altera SoC FPGA Device
HPS Portion
FPGA Portion
Flash Controllers
SDRAM Controller Subsystem
Cortex-A9 MPU Subsystem
On-Chip Memories
Support Peripherals
PLLs
Interface Peripherals
Debug
HPS-FPGA Interfaces
Control Block
User I/O
HSSI Transceivers
FPGA Fabric (LUTs, RAMs, Multipliers & Routing)
PLLs
Hard PCIe
Hard Memory Controllers

FIGURE 2
Altera SoC FPGA Device's block diagram [1].

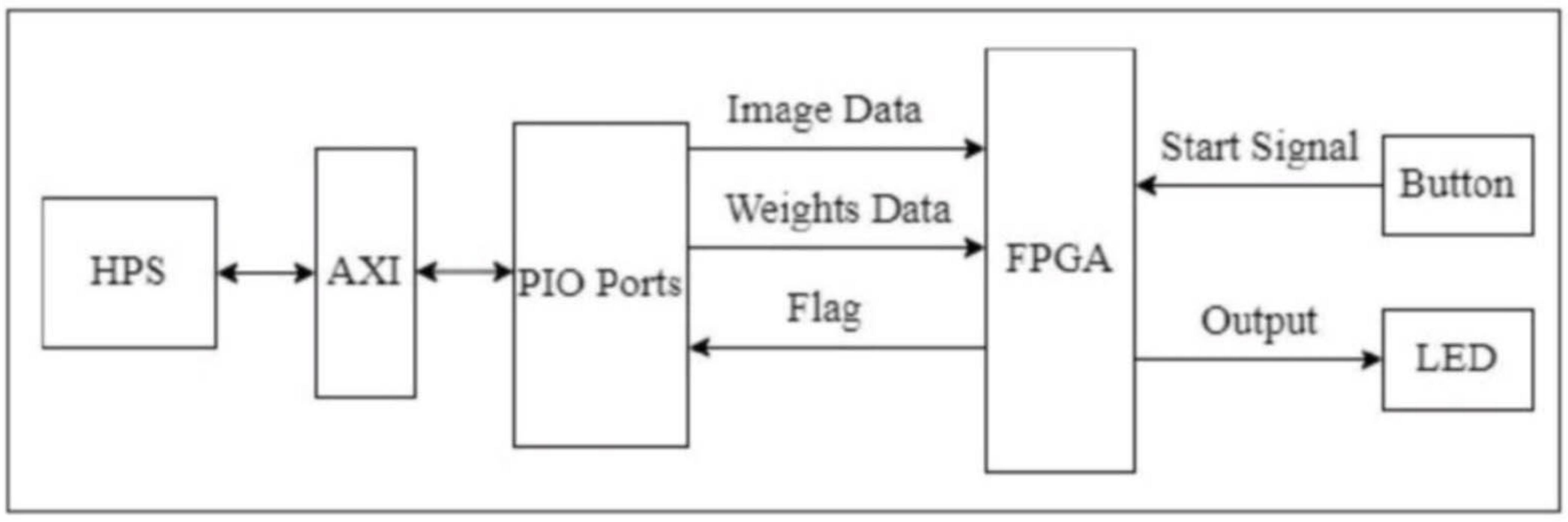


FIGURE 3
Communication demonstration between HPS and FPGA.

INTRODUCTION TO DE1-SOC

As shown in **Figure 2**, the DE1-SoC Development Kit is a hardware platform built around the Altera System-On-Chip (SoC) FPGA, which integrates two key components: the Hard Processor System (HPS) and the FPGA. The HPS contains two ARM Cortex-A9, handling general-purpose tasks, while the FPGA is used for custom hardware processing, allowing for efficient, high-speed computations. Together, these components provide a flexible and powerful design environment for tasks such as real-time image recognition and other advanced processing applications.

The HPS communicates with the FPGA by means of shared memory. Data transfer from the HPS to the FPGA is handled through memory mapping. On the FPGA side, communication happens via the AXI bus, which is an interface protocol that enables efficient communication between different components in a SoC. By defining the addresses for parallel ports (referred to as "pio ports") and ensuring both sides are properly synchronized, we enable seamless data exchange between the HPS and FPGA. As shown in **Figure 3**, this setup facilitates smooth interaction between the two components in our design.

TRAINING OUR CNN MODEL WITH PYTORCH

Before we could run our handwritten digit recognition system on hardware, we first had to test its performance using software and obtain the necessary weights for each layer of the system. We used PyTorch to generate and test our CNN weights. Instead of manually coding all the algorithms, we just needed to specify some attributes for CNN in PyTorch, and it automatically built the neural network for us. We trained our model using a portion of the MNIST dataset (a popular collection of handwritten digits) and tested it with another portion, achieving 99% accuracy. Afterward, we extracted the weights for both the convolutional and fully connected layers, which will be used when deploying the system on hardware. PyTorch also made it simple to gather these weights.

SYSTEM DESIGN

The C program running on the HPS manages data transfer between our computer and the handwriting recognition system. For input, we used images of digits drawn in Microsoft Paint.

Once the image is preprocessed by our C program, it gets sent to the FPGA. At the same time, the weights of the convolutional layer are also sent to the FPGA. The system starts when we press the built-in SoC button to issue a start signal to the FPGA. The FPGA then processes the image through the network. However, due to memory constraints—only three memory blocks are allocated—the FPGA halts processing just before the fully connected layer and sends a signal back to the HPS. This pause allows for memory swapping, where weights from the convolutional layer are replaced with the one from the fully connected layer to make room for subsequent computations.

After receiving the needed data, the FPGA completes the process and gives us 10 numbers. The index of the number with the highest value is the result. While we could apply a "softmax" function to calculate probabilities for each number, it's not necessary during recognition—it's mainly useful when training the system.

CNN DESIGN IN FPGA

In our convolutional layer design, we use one multiplier for each convolutional unit, which is the minimum component to conduct convolutional arithmetic. The mathematical expression for the convolution output element at position (*i,j*) is given by:

$$y_{i,j} = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x_{i+m,j+n} \cdot w_{m,n}$$

where: $y_{i,j}$ is the output value at position (*i,j*) in the output feature map. *x* represents the

ABOUT THE AUTHORS

Jichao Yang (jy874@cornell.edu) was master student at Cornell University, studying at Electrical and Computer Engineering. He finished his degree in December of 2024. He will be joining Apple as an RF validation engineer after graduation. He is interested in microwave designs and IoT devices.

Yiyang Zhao (yz2952@cornell.edu) was is an M.Eng. student at Cornell University majoring in Electrical and Computer Engineering. He finished his degree in December of 2024. His passion lies at the intersection of hardware and software, with a focus on areas such as operating systems, embedded software, and hardware-software co-design.

input feature map. *w* represents the kernel (filter) weights. *b* is the bias term (optional, depending on the architecture). *M* and *N* are the height and width of the kernel. The sums are performed over all elements within the kernel window.

As illustrated in **Figure 4**, at least nine clock cycles are needed to perform the convolutional arithmetic. It starts with the top left pixel in the kernel. In the next cycle, move to the right to the next pixel for multiplication, and then accumulate with the previous value. This process is repeated until the entire convolutional unit is calculated, and then outputs the accumulated value. Considering the state of idle and storing the data, we need 14 clock cycles to finish a convolutional calculation since each convolutional result should be put into the SRAM at a time.

In the convolutional layer, the arithmetic operations begin at the top-left corner of the image. The convolutional kernel then moves across the image, processing each 3x3 section one at a time. This process is repeated until the entire image has been covered, ending at the bottom-right corner. Once complete, the outputs from the convolutional layer are stored in the designated SRAM. To speed up the process, we implemented three parallel channels for concurrent processing. The control flow within the convolutional layers is managed by a state machine illustrated in **Figure 5**, ensuring seamless operation. In the IDLE state, image data and weights for both the convolutional and fully connected layers are loaded. The CALC state handles the computation for each convolutional unit, while the STORE state saves the computed output to SRAM and shifts the kernel to the next computation unit. Finally, the DONE state signals the max-pooling layer to begin its processing.

In our design of the max-pooling layer module, we imitated the convolutional layer control flow. The difference is that the internal state machine is quite simple, because the maximization operation can be done without complex logic. In the IDLE state, it waits for the start signal, which is also the end signal of the convolutional layer. It transits to the CALC state when it receives that signal.

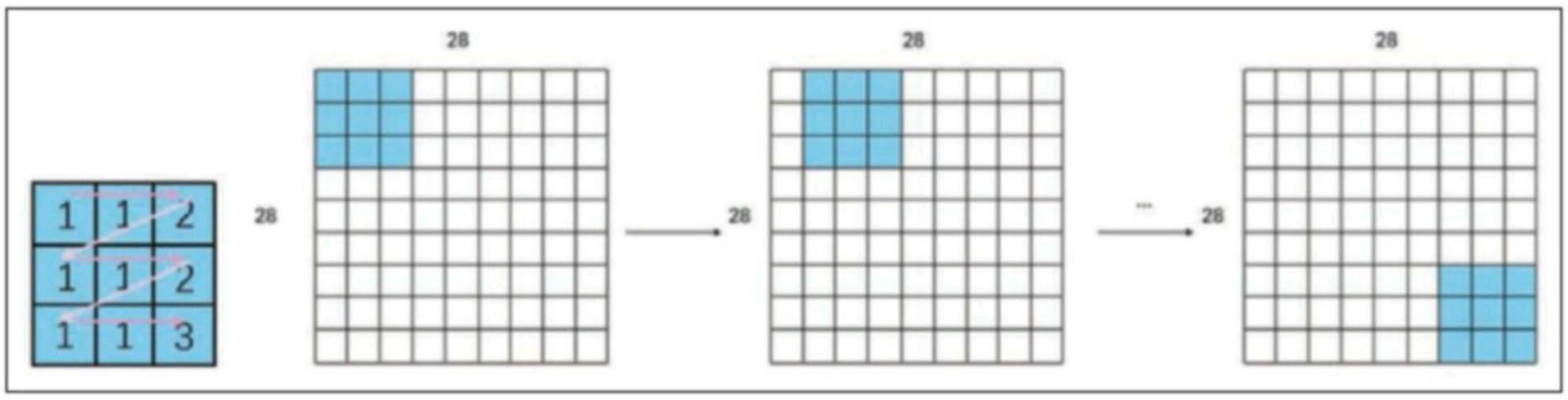

1 1 2
1 1 2
1 1 3
28
28
28
28
...
28
28

FIGURE 4
Convolutional unit and arithmetic process of convolutional layer.

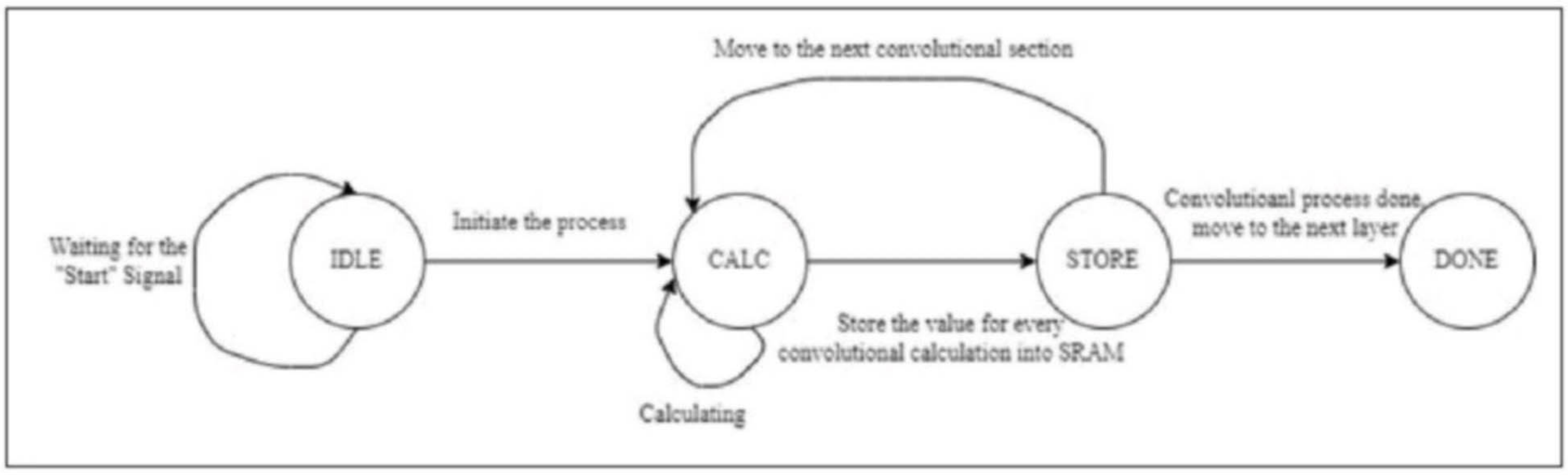

Move to the next convolutional section
Waiting for the "Start" Signal
IDLE
Initiate the process
CALC
STORE
Convolutioanl process done, move to the next layer
DONE
Store the value for every convolutional calculation into SRAM
Calculating

FIGURE 5
State machine for the convolutional layer.

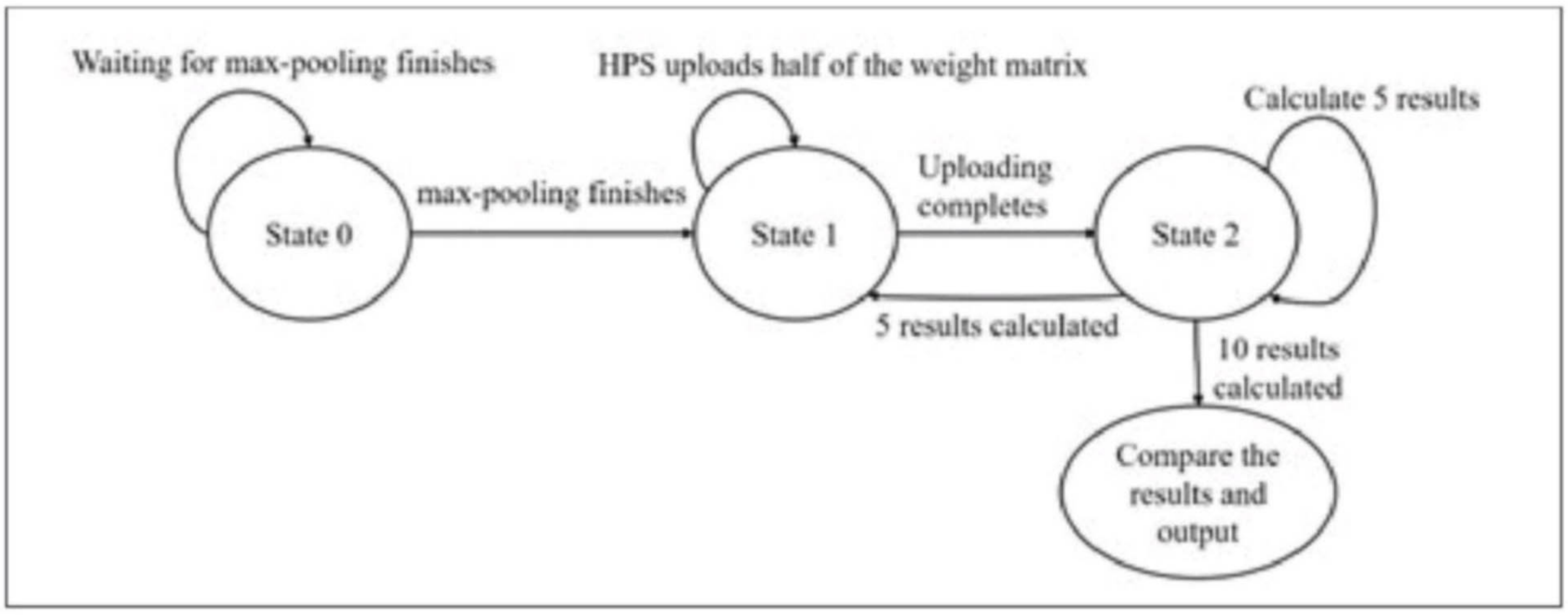

Waiting for max-pooling finishes
HPS uploads half of the weight matrix
Calculate 5 results
State 0
max-pooling finishes
State 1
Uploading completes
State 2
5 results calculated
10 results calculated
Compare the results and output

FIGURE 6
Computation flow for the fully connected layer.

Next, it traverses the input matrix to find the maximum value in each 2×2 subregions. After that, the state transits into WRITE state and the maximum value will be stored into the appropriate destination SRAM address. Finally, it reaches the DONE state and gets ready for the next subregion processing.

The fully connected layer converts the 507 outputs from the max pooling layer into 10 output values. The index, which contains the highest output value, is the predicted number. However, As the weight matrix has a size of 507×507, it is not possible to upload the matrix to SRAM at one time because of limited memory. Our solution is to upload the matrix and calculate the output two times, as illustrated in the state machine in **Figure 6**. The final predicted number of the CNN will be shown in the digital display.

RESULTS

Some of the results are shown here. The leftmost LED represents the output.

In **Figure 7** and **Figure 8**, we show our system in action, successfully recognizing the handwritten digits 3 and 7. To put our design to the test, we used Microsoft Paint to draw 10 different images of each digit (0-9) and ran them through our recognition system. The results are summarized in **Table 1**, where you can see the accuracy for each digit.

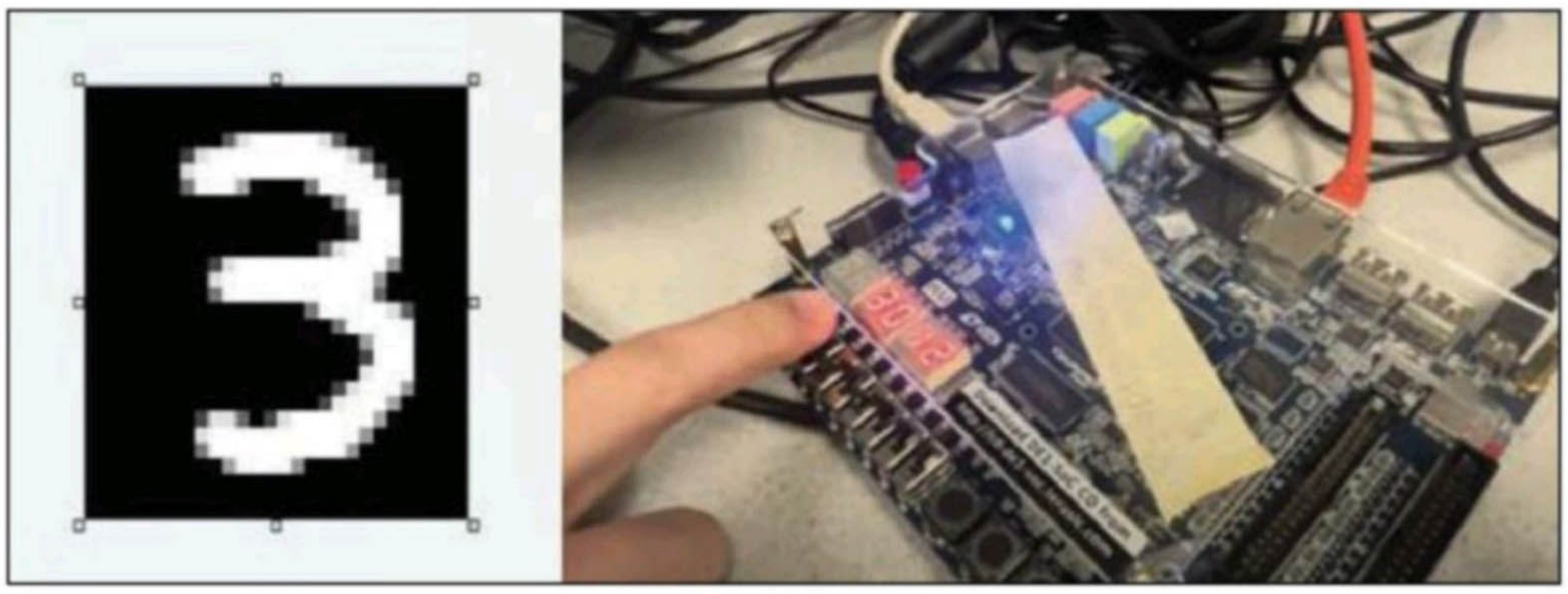


FIGURE 7
Test result for Digit 3 (left: handwritten input; right: output).

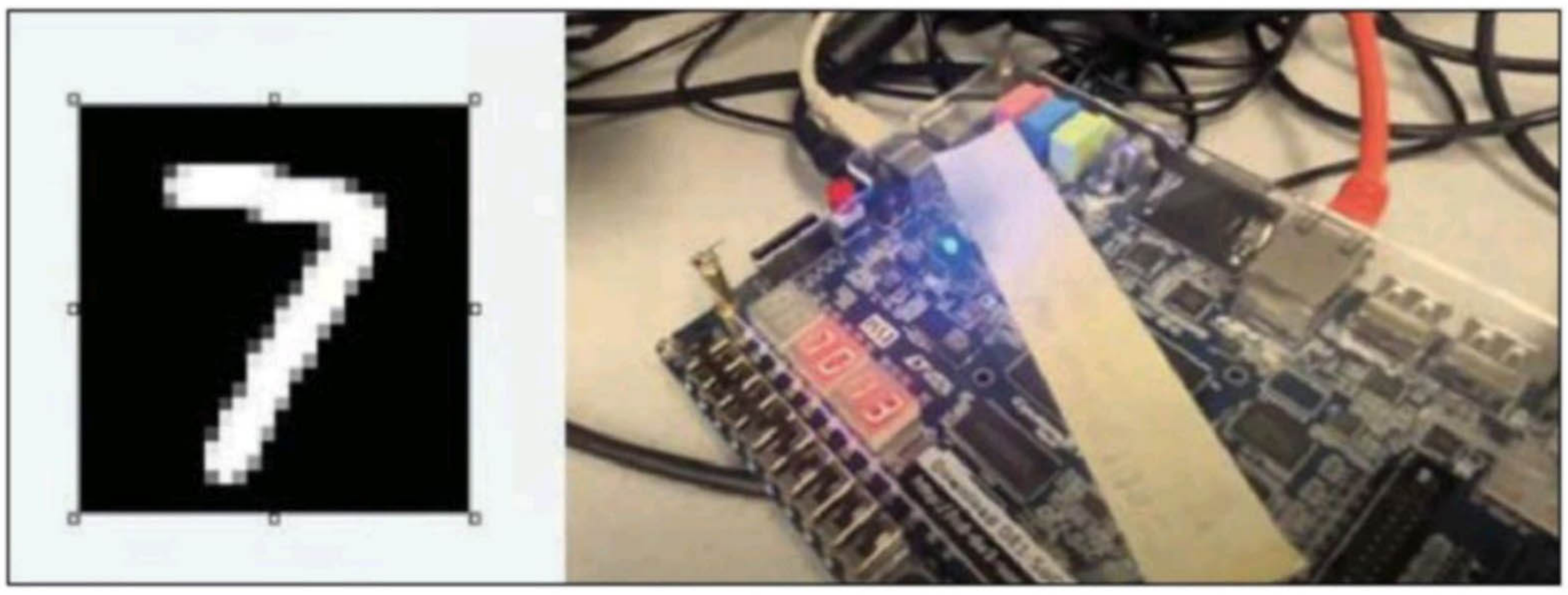


FIGURE 8
Test result for Digit 7 (left: handwritten input; right: output).

Number	FPGA Accuracy	PyTorch Accuracy
1	90%	100%
2	100%	100%
3	90%	100%
4	100%	100%
5	70%	90%
6	80%	100%
7	90%	100%
8	100%	100%
9	90%	100%
0	100%	100%
Average	91%	99%

TABLE 1
FPGA and PyTorch accuracy for recognition, each number tested 10 times.

Overall, our system's average accuracy is a bit lower than what we get from PyTorch. One reason for this difference could be the numerical formats: PyTorch uses single-precision floating-point numbers, which are more precise, while our FPGA system operates on 16-bit fixed-point numbers. This difference in precision might cause some slight variations in performance. One approach to resolving this issue is to maintain consistency in the numerical format between PyTorch and the FPGA, which is very likely to enhance accuracy. We might also use Quantization Aware Training provided by PyTorch, enabling us to simulate fixed-point effects during training and convert the trained model to a quantized fixed-point representation on Verilog.

CONCLUSION

We've documented the results of our project in a detailed video [2]. Our innovative design gives the CNN recognition algorithm a serious speed boost, with our FPGA-powered setup reaching an impressive 91% accuracy. Of course, there's always room for further improvement—such as hooking up a real-time camera, switching from fixed-point to floating-point math, or training a fancier model in PyTorch for better weight matrices and performance.

If we're feeling adventurous, we could complicate things even further by throwing in a few extra layers between the input and output. But hey, why make things simple when you can make them complex, right?

Additional materials from the author are available at:
www.circuitcellar.com/article-materials

References [1] to [2] as marked in the article can be found there.

RESOURCES

Altera | www.altera.com
Intel | www.intel.com

FREE Download Now!
Mac, Windows and Linux

Blackmagicdesign

DaVinci Resolve
File
Edit
Trim
Timeline
Clip
Mark
View
Playback
Fusion
Color
Fairlight
Workspace
Help
Media Pool
Sync Bin
Transitions
Titles
Effects
Aerospace Propulsion Project
Quick Export
Full Screen
Inspector
Master / Clips
Main Edit
01:00:12:17
DaVinci Resolve 19
Media
Cut
Edit
Fusion
Color
Fairlight
Deliver

Edit and color correct using the same software used by Hollywood, for free!

DaVinci Resolve is Hollywood's most popular software! Now it's easy to create feature film quality videos by using professional color correction, editing, audio and visual effects. Because DaVinci Resolve is free, you're not locked into a cloud license so you won't lose your work if you stop paying a monthly fee. There's no monthly fee, no embedded ads and no user tracking.

Editing, Color, Audio and Effects!
DaVinci Resolve is the world's only solution that combines editing, color correction, visual effects, motion graphics and audio post production all in one software tool! You can work faster because you don't have to learn multiple apps or switch software for different tasks. For example, just click the color page for color, or the edit page for editing! It's so incredibly fast!

Professional Editing
DaVinci Resolve is perfect for editing sales or training videos! The familiar track layout makes it easy to learn, while being powerful enough for professional editors. You also get a library full of hundreds of titles, transitions and effects that you can add and animate! Plus, DaVinci Resolve is used on high end work, so you are learning advanced skills used in TV and film.

Creative Color Correction
DaVinci Resolve's color page is Hollywood's most advanced color corrector and has been used on more feature films and television shows than any other system! It has exciting new features to make it easier to get amazing results, even while learning the more advanced color correction tools. There's PowerWindows™, qualifiers, tracking, advanced HDR grading tools and more!

Designed to Grow With You
DaVinci Resolve is designed for collaboration so as you work on larger jobs you can add users and all work on the same projects, at the same time. You can also expand DaVinci Resolve by adding a range of color control panels that let you create unique looks that are impossible with a mouse and keyboard. There's also edit keyboards and Fairlight audio consoles for sound studios!

DaVinci Resolve 19	**Free**
DaVinci Resolve Micro Color Panel	**Only $509**

Learn the basics for free then get more creative control with our accessories!

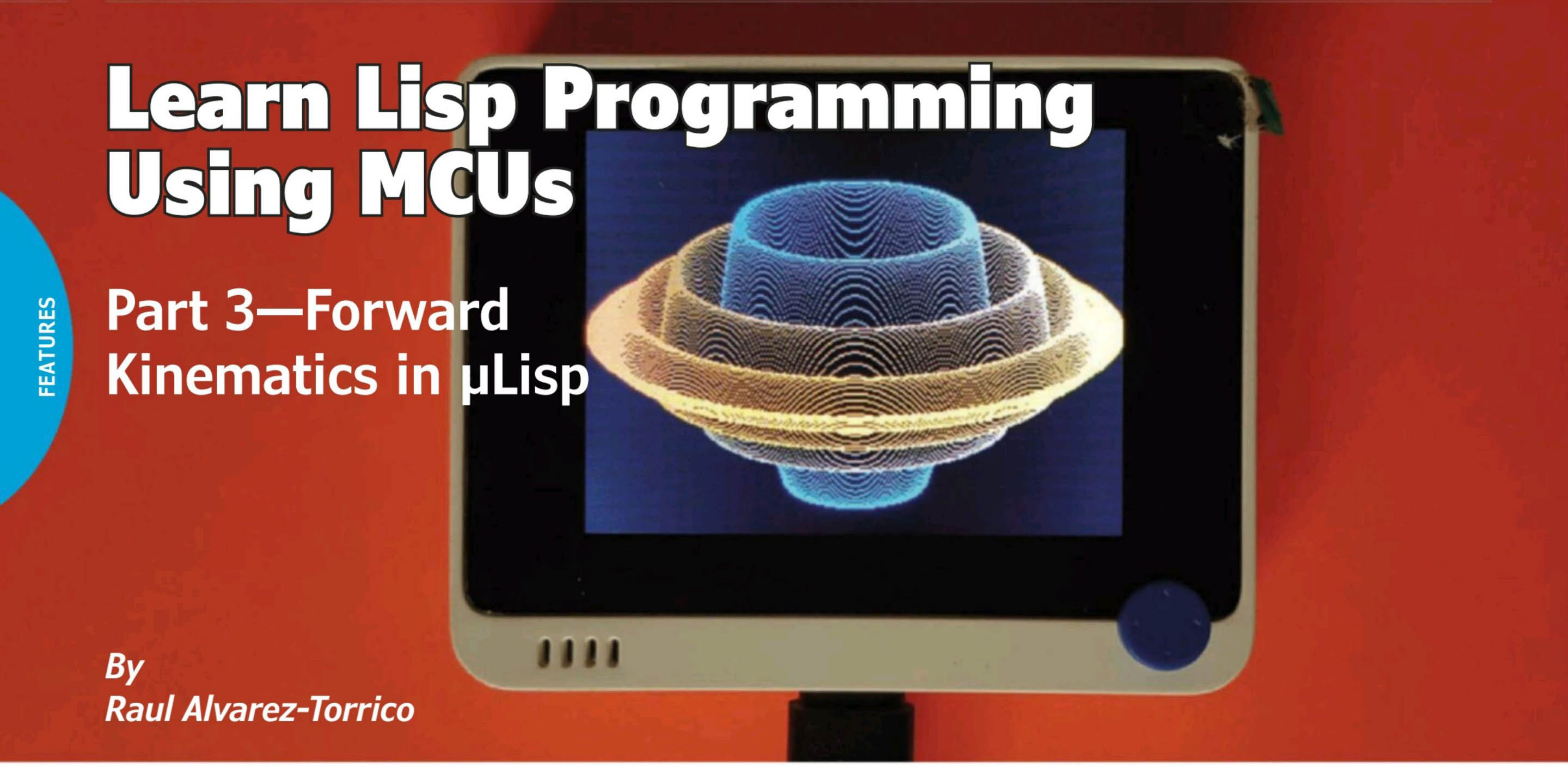
⇢ **Learn more at www.blackmagicdesign.com**

Download free on the
DaVinci Resolve website

NO SUBSCRIPTIONS • NO ADS • NO USER TRACKING • NO AI TRAINING

Learn Lisp Programming Using MCUs

Part 3—Forward Kinematics in µLisp



By
Raul Alvarez-Torrico

µLisp is a dialect of the Lisp programming language for MCUs with limited RAM. In Part 1, Raul presented µLisp concepts and syntax, and extensions specific to MCUs, and in Part 2, he added extensions by using custom functions. In Part 3 of this article series, he implements a new extension to multiply matrices, and uses µLisp to implement forward kinematics for a mini robotic arm.

In this last part of the series on Lisp programming with microcontrollers, we will use everything we previously learned about µLisp programming. We will compute "forward kinematics" (the process of computing the pose of a robotic arm, based on the angles of its rotating joints), using homogeneous transformations for the three-degrees-of-freedom (3DOF) mini robotic arm presented in Part 2 of this article series [1]. To achieve this, we will implement a matrix multiplication extension to µLisp. Along the way, we will also continue to further explore the Lisp programming language, while we challenge ourselves with math and the fundamental concepts of robotic arm kinematics.

To follow the topics discussed here, you need to be familiar with Arduino and µLisp programming. Please see the first two parts of this article series for information [1, 2]. "Robot kinematics" refers to the relation between the geometry and movement of a robot. It is also desirable to have some background in robot kinematics with the Denavit-Hartenberg convention and homogeneous transformations, so I will give a simple explanation of the basics. The latter, however, is really tangential to the subject of the entire article series.

LISTS AND ARRAYS

Lists are one of the most fundamental data structures in Lisp. In Part 1, I discussed how to create them. Lines 1-2 from **Listing 1** are a reminder of two equivalent methods discussed. However, there's a third method of creating lists by "consing" atoms (Lisp primordial objects) with the `cons` function. For example, line 4 shows how to create one with just one element. Because `cons` always takes two arguments, we put `nil` as the second one. We can also cascade two `cons` to obtain the two-element list shown in lines 7-8. Here, the inner `cons` builds the same one-element list from line 5. The outer `cons` appends "7" to the previous list to obtain the two-element list shown in line 8. In a similar fashion, line 10 builds the four-element list shown in line 11. Now you may be thinking that consing is not really practical if we obtain the same result using the shorter methods from lines 1-2. The answer is that in some cases, consing is really useful for specific purposes.

Now, line 13 shows how to create a list and bind it to the variable `lst`. Afterward, we can use the `progn` Lisp function to inspect its contents (lines 15-16). Furthermore, the `car` Lisp function applied to the same list returns its first element, which is "5" (lines

18-19). Conversely, the `cdr` function returns the rest of the list without the first element (lines 21-22). There are also the functions `first`, `second`, and `third`, which return list elements in those corresponding positions (lines 24-28). Now, applying the `cdr` function to the list twice in cascade returns the third and fourth elements as a sub-list (lines 30-31). The first `cdr` returns "(6 7 8)" and the second `cdr` applied to the latter returns "(7 8)." Because this is a common operation with lists, there's the equivalent function `cddr` that does the same (line 33). The double "dd" in the name means two cascading `cdr`s.

Now let's talk about how to create arrays. Line 37 shows the S-expression to create a one-dimensional (1D) array using the `make-array` function. The first argument—"4"—is the number of array elements, and the `:initial-elements` keyword symbol specifies their initial value—which is "0." Line 38 shows the result. The symbol "#" in front of the parenthesis is a "reader macro" that introduces the 1D array. Line 40 shows the creation of a two-dimensional (2D) array, and line 41 shows the result with the corresponding "#2A" reader macro. To create the same array and assign it to a variable `arr`, we use the S-expression in line 45.

Once we have this variable, we can access individual values with the function `aref`. For instance, line 48 accesses the element with subscripts "(2 3)." That is, the element in the third row and fourth column—because indexing in Lisp starts at "0." The returned value is "0" because all elements were initialized to that value. But we can use the same referencing scheme along with the `setf` function to change individual element values as shown in line 51. Here we are changing the value of element "(2 3)" to "50." So, reading it again (line 54) now returns "50." If we want to check the whole array content, we can apply the `eval` function to it (lines 57-58). Applying the `progn` function produces the same effect.

More details regarding the use of lists, arrays, and other functions are available on the μLisp language reference webpage [3].

ROBOTIC ARM KINEMATICS

"Robot kinematics" refers to the relation between the geometry and the movement of a robot. For a robotic arm with rotating joints, it's the relation between the joint rotation angles and the spatial coordinates of its link-ends, in particular the coordinates of the end effector (the claw).

In that vein, "forward kinematics" generally refers to the process of computing the pose

```
1 (list 5 6 7 8)
2 ‘(5 6 7 8)
3
4 1929> (cons 8 nil)
5 (8)
6
7 1929> (cons 7 (cons 8 nil))
8 (7 8)
9
10 1929> (cons 5 (cons 6 (cons 7 (cons 8 nil))))
11 (5 6 7 8)
12
13 (defvar lst (list 5 6 7 8))
14
15 1800> (progn lst)
16 (5 6 7 8)
17
18 1830> (car lst)
19 5
20
21 1830> (cdr lst)
22 (6 7 8)
23
24 1830> (first lst)
25 5
26
27 1830> (second lst)
28 6
29
30 1929> (cdr (cdr lst))
31 (7 8)
32
33 1929> (cddr lst)
34 (7 8)
35
36 ;; Arrays
37 1804> (make-array '(4 4) :initial-element 0)
38 #(0 0 0 0)
39
40 1830> (make-array 4 :initial-element 0)
41 #2A((0 0 0 0) (0 0 0 0) (0 0 0 0) (0 0 0 0)
42
43 (defvar arr (make-array ‘(4 4) :initial-element 0))
44
45 1830> (defvar arr (make-array ‘(4 4) :initial-element 0))
46 arr
47
48 1805> (aref arr 2 3)
49 0
50
51 1805> (setf (aref arr 2 3) 50 )
52 50
53
54 1805> (aref arr 2 3)
55 50
56
57 1804> (eval arr)
58 #2A((0 0 0 0) (0 0 0 0) (0 0 0 50) (0 0 0 0))
```

LISTING 1
List and array indexing in μLisp.

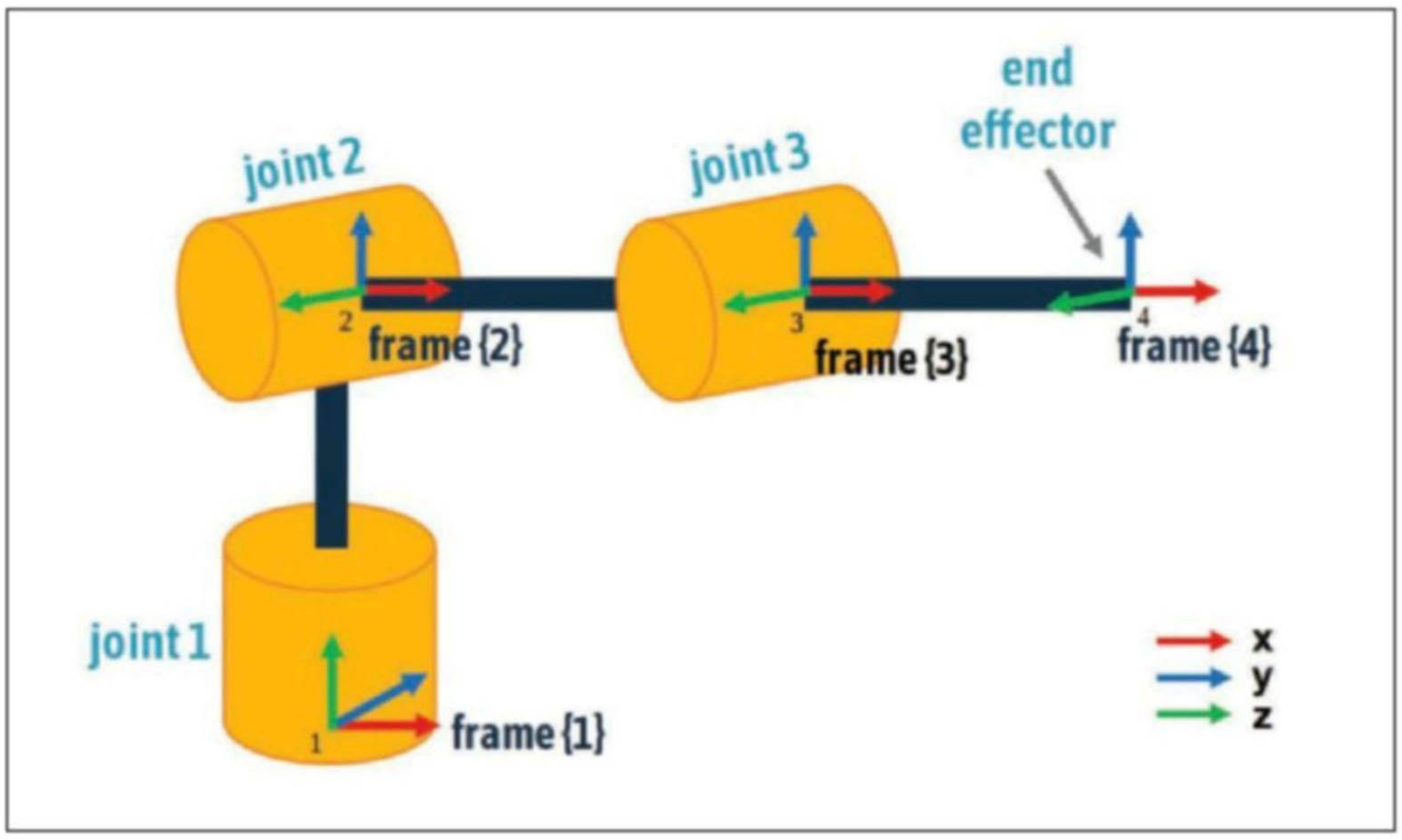
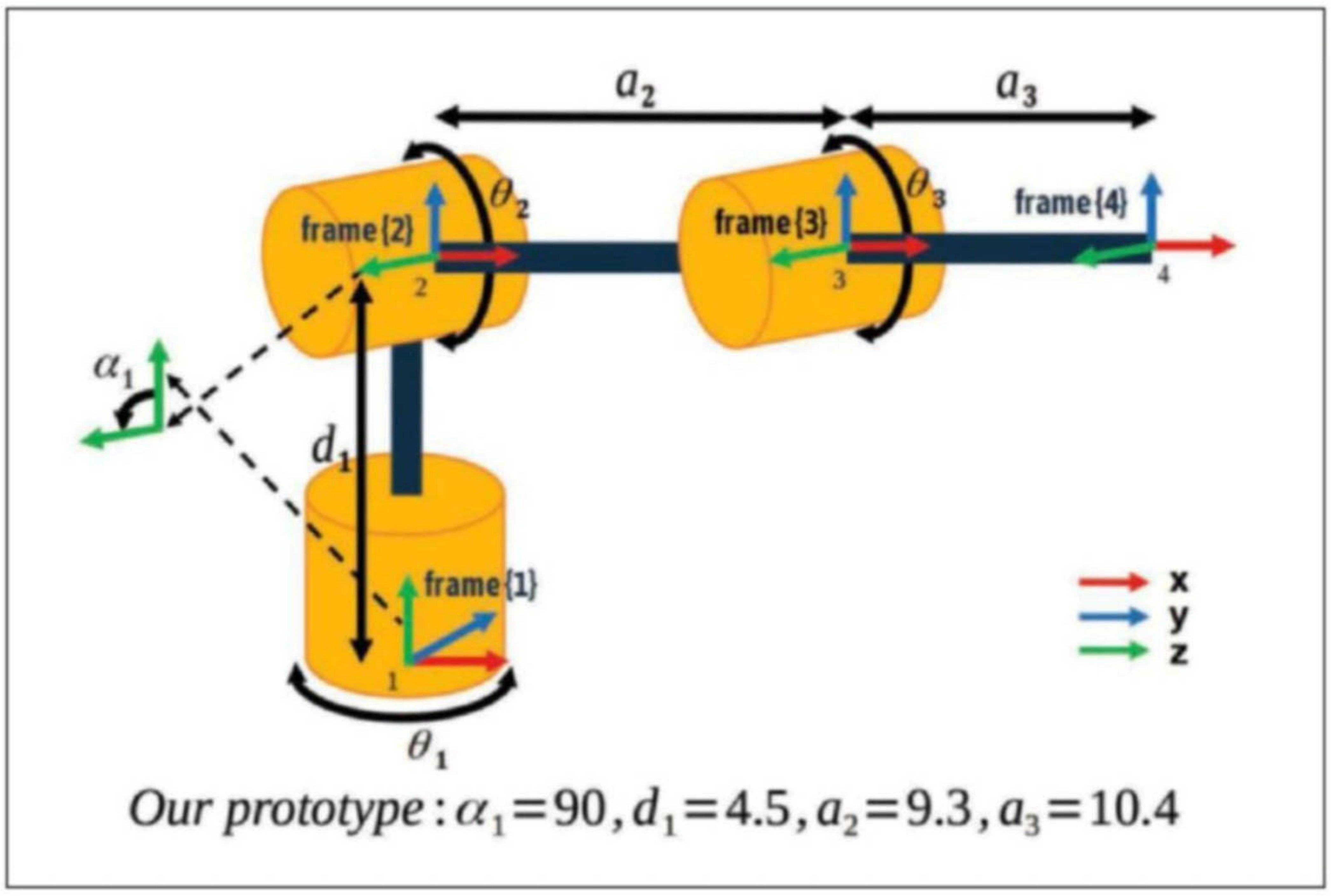
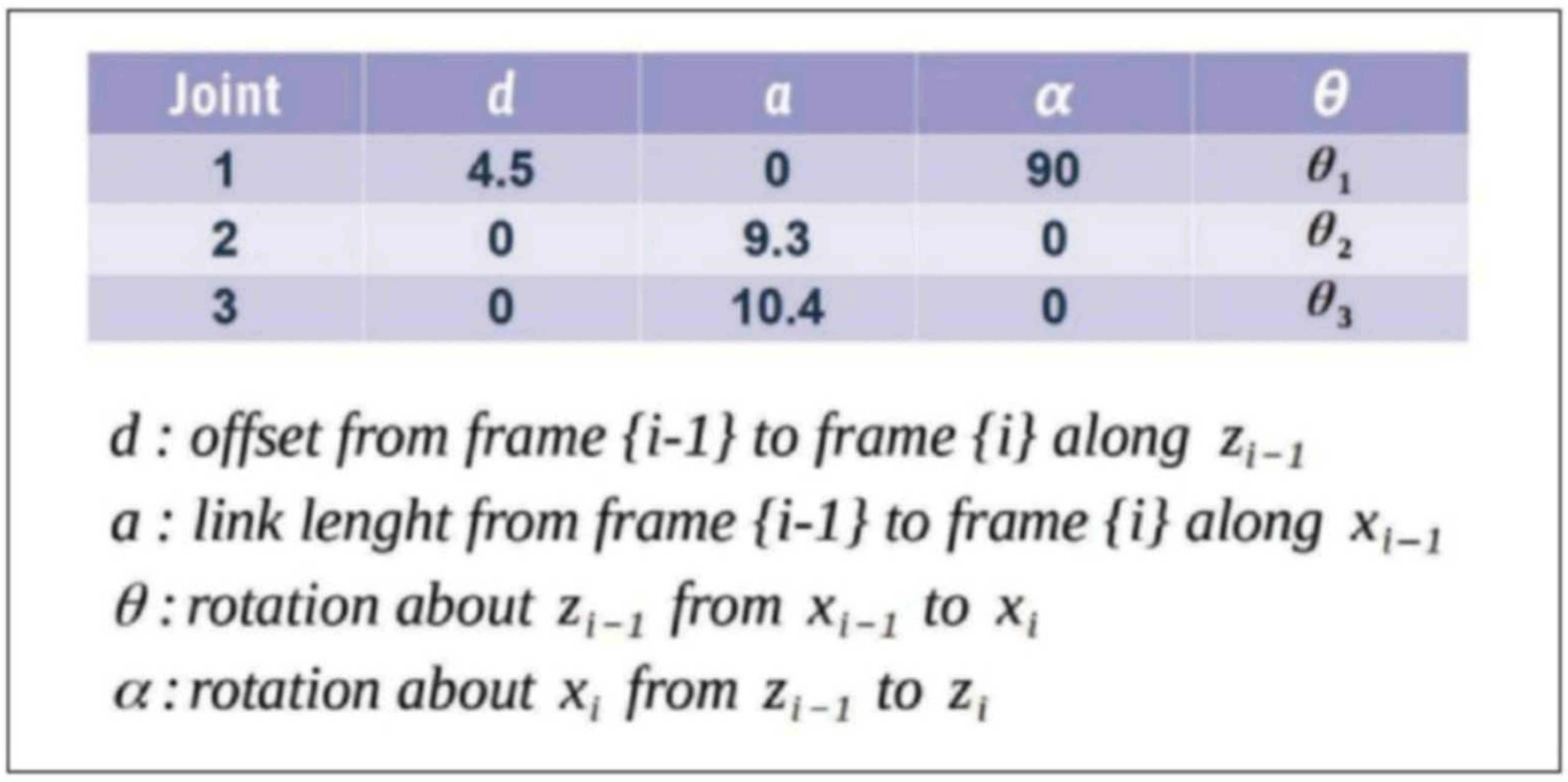


FIGURE 1
3DOF robotic arm with assigned coordinate frames.



Our prototype: $\alpha_1=90, d_1=4.5, a_2=9.3, a_3=10.4$

FIGURE 2
Denavit-Hartenberg parameters for a 3DOF robotic arm.



Joint	d	a	α	θ
1	4.5	0	90	θ_1
2	0	9.3	0	θ_2
3	0	10.4	0	θ_3

d : offset from frame {i-1} to frame {i} along z_{i-1}
a : link lenght from frame {i-1} to frame {i} along x_{i-1}
θ *: rotation about* z_{i-1} *from* x_{i-1} *to* x_i
α *: rotation about* x_i *from* z_{i-1} *to* z_i

TABLE 1
DH parameter table.

of the end effector—and all link-ends—as a function of the joint rotation angles. This can be mathematically expressed as:

$$X = f(\Theta) \quad [1]$$

Conversely, "inverse kinematics" is the process of computing the joint angles as a function of the end effector pose:

$$\Theta = f^{-1}(X)$$

where "pose" really means position and orientation. For simplicity, we will only take into account position in 3D space.

Arguably, one of the most well-known methods to compute forward and inverse kinematics is the one that uses homogeneous transformations (HT) with the Denavit-Hartenberg (DH) convention. This is the method we will use here. Now, this is a whole topic in and of itself, so I can't talk about it in detail. For our purposes, I will discuss the most basic concepts to provide a suitable introduction to the topic for those not familiar with it.

DH kinematics involve the concept of "homogeneous transformations," which in turn involve matrix multiplications. What's more, the Denavit-Hartenberg convention implies a set of rules to establish the so-called Denavit-Hartenberg parameters for each link in the robotic arm. These DH parameters help us build the HT matrices that express an absolute relation between joint angles and link-end positions. Although conceptually a little hard to grasp due to the fair amount of math involved, this is, in fact, very easy to implement in code, using a linear algebra library that includes matrix operations.

A homogeneous transformation is just a matrix multiplication that helps find the pose of a joint or link-end with respect to a given reference coordinate frame. Examples are the position of coordinate frame {2} in **Figure 1**—which is the end of the first "link"—with respect to frame {1}, which is the global or world reference frame; or the position of frame {3} with respect to frame {2}, and so on until reaching the end effector, which is frame {4}. Thus, the DH convention also establishes a way to attach reference frames to link-ends, in order to apply homogeneous transformations. **Figure 2** is the graphical representation of the four DH parameters in a 3DOF robotic arm. For such parameters—given in **Table 1**—the DH convention states the following rules:

1) Coordinate frames must be attached to joints between consecutive links, including the first and last link-ends.

2) For each coordinate frame, the z-axis (the green arrows in Figure 2) is in the joint rotation axis direction. For instance, joint 1 rotates with a angle around the first vertical green z-axis, and so on for the rest of the joints.
3) The x-axis is found as the vector product: $x_i = z_i \times z_{i-1}$. That is, between the current z-axis and the previous z-axis. As it is the case for the pair of joints 2 and 1, which z-axes form a 90° angle. If the current z-axis is parallel to the previous z-axis, the x-axis is along the common normal between the two, going from the previous to the current. It is the case for the pair of joints 3 and 2, and also for the end effector and joint 3.
4) Find the y-axes by following the right-hand rule.
5) Once coordinate frames are attached to joints, the four DH parameters shown in Table 1 are determined as follows:

- Joint angle is the joint rotation angle around the z-axis. See in Figure 2.
- Joint distance "d" is the distance from one coordinate frame to the next along the z-axis. For instance, there's "d1" from frame {1} to frame {2}. But between frames {2} and {3} there's no distance along the z-axis, so "d2" is zero. Distance "d3" is also zero for the next pair of frames {3} and {4}.
- Link length "a" is the distance from one frame to the next along the x-axis. For instance, there's a distance "a2" from frame {2} to frame {3}, which is effectively the length of link 2. There's also "a3" from frame {3} to frame {4} at the end effector, which is the link 3 length. From frame {1} to frame {2}, however, there's no distance along the x-axis, so in this case, "a1" is zero. Even if intuitively it appears to be a "link length" between frames {1} and {2}, this is really "d1", which is technically a "displacement" along the z-axis, not a link length.
- Link twist angleα: is the rotation angle of a frame from the previous z-axis to the next z-axis around the next x-axis. For instance, between frames {2} and {1} there's an angle of 90°. This means that if you align your right thumb with the red arrow (x-axis) in frame {2} and your other four fingers with the green arrow (z-axis) in frame {1}, you must rotate your four fingers around your thumb +90° to align them with the green arrow {z-axis) in frame {2}. So, α_1 is 90°. Between frames {3} and {2}, the rotation angle is zero, because their z-axes are already aligned, so α_2 is zero. Between frames {3} and {4}, α_3 is zero as well.

Table 1 contains values of the four parameters for all joints and frames.

Once we have established all DH parameters, we can specify the HT matrices that define the relation between each coordinate frame and the next. The following is the general formulation of the DH transformation matrix:

$$ {}^{i\to 1}T_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

To find the transformation matrix from frame {1} to frame {2}, we just fill the values from row 1 in Table 1:

$$ {}^{1}T2 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1)\cos(90) & \sin(\theta_1)\sin(90) & 0\times\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1)\cos(90) & -\cos(\theta_1)\sin(90) & 0\times\sin(\theta_1) \\ 0 & \sin(90) & \cos(90) & 4.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

For the transformation from frame {2} to frame {3} we have:

$$ {}^{2}T_3 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2)\cos(0) & \sin(\theta_2)\sin(0) & 9.3\times\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2)\cos(0) & -\cos(\theta_2)\sin(0) & 9.3\times\sin(\theta_2) \\ 0 & \sin(0) & \cos(0) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

And finally, for the transformation from frame {3} to frame {4}—that is, the end-effector:

$$ {}^{3}T_4 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3)\cos(0) & \sin(\theta_3)\sin(0) & 10.4\times\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3)\cos(0) & -\cos(\theta_3)\sin(0) & 10.4\times\sin(\theta_3) \\ 0 & \sin(0) & \cos(0) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Now, to obtain the end-effector pose with respect to the world coordinate frame, which will generally be frame {1}, we perform the following matrix multiplications:

$$ {}^{1}T_4 = {}^{1}T_2 \times {}^{2}T_3 \times {}^{3}T_4 $$

And ${}^{1}T_4$ is the homogeneous transformation matrix that describes the end-effector pose relative to the origin of the world coordinate system. These matrices give us essentially all we need to compute the robotic arm's forward kinematics.

I won't elaborate further on the DH convention and homogeneous transformations, but you can find additional resources on this topic on the *Circuit Cellar* Article Materials and Resources webpage.

µLISP MATRIX MULTIPLICATION

Based on what we have discussed, to do forward kinematics in µLisp, we obviously need at least a matrix multiplication function. This is not available by default in µLisp, but can be implemented as an extension.

In Part 2 of this article series [1] I explained how to add extensions to µLisp. **Listing 2** shows the C language code for the µLisp `mat-mult` matrix multiplication function. For simplicity sake we are ignoring the auxiliary code needed for the µLisp extension system, but the full source code is

ABOUT THE AUTHOR

Raul Alvarez-Torrico has a BEng in electronics and is the founder of TecBolivia, a company offering services in physical computing and educational robotics in Bolivia. In his spare time, he likes to experiment with wireless sensor networks, robotics, and artificial intelligence. He is also committed to publishing articles and video tutorials about embedded systems and programming in his native language (Spanish), at their company's site www.TecBolivia.com. You may contact him at raul@tecbolivia.com.

available for download on the *Circuit Cellar* Article Code and Files webpage.

Keeping aside the most trivial details, let's see how the code works. Once the function is compiled and added to μLisp, it will be possible to call it as: `(mat-mult a b)`, where "a" and "b" are the two 2D matrices to multiply. Matrices that will be defined in μLisp as 2D arrays—which are analogous to 2D arrays in C. So, the two matrices are passed as arguments to the `fn_matmult()` C function in Listing 2. These are technically 2D μLisp array objects, which means they must be processed in C accordingly. μLisp array objects are explained in detail on the μLisp project website [4].

```
1 object *fn_matmult (object *args, object *env) {
2  (void) env;
3  int bit; int rows, cols;
4  object *resarray;
5
6  int nargs = listlength(args);
7  if (nargs == 2) {
8    object *array1 = first(args);
9    object *dims = cddr(array1);
10    rows = first(dims)->integer;
11    cols = second(dims)->integer;
12
13    mtx_type A[rows][cols];
14
15    for(int i = 0; i < rows; ++i){
16     for(int j = 0; j < cols; ++j) {
17      object *rowcol = cons(number(i), cons(number(j), NULL));
18      object **loc = getarray(array1, rowcol, 0, &bit);
19      A[i][j] = checkintfloat(*loc);
20     }
21    }
22
23    object *array2 = second(args);
24    dims = cddr(array2);
25    rows = first(dims)->integer;
26    cols = second(dims)->integer;
27
28    mtx_type B[rows][cols];
29
30    for(int i = 0; i < rows; ++i){
31     for(int j = 0; j < cols; ++j) {
32      object *rowcol = cons(number(i), cons(number(j), NULL));
33      object **loc = getarray(array2, rowcol, 0, &bit);
34      B[i][j] = checkintfloat(*loc);
35     }
36    }
37
38    mtx_type R[rows][cols];
39    Matrix.Multiply((mtx_type*)A, (mtx_type*)B, rows, cols,
cols, (mtx_type*)R);
40
41   object *resdims = cons(number(rows), cons(number(cols), NULL));
42    resarray = makearray(resdims, number(0), false);
43
44    for(int i = 0; i < rows; ++i){
45     for(int j = 0; j < cols; ++j) {
46      object *rowcol = cons(number(i), cons(number(j), NULL));
47      object **loc = getarray(resarray, rowcol, 0, &bit);
48        *loc = makefloat(R[i][j]);
49     }
50    }
51  } else error2(PSTR(“wrong number of arguments”));
52  return resarray;
53 }
```

LISTING 2
Code for the μLisp "mat-mult" extension.

After checking that the function has received exactly two arguments (line 7), we proceed to obtain a pointer to the first one (line 8). This pointer points to a list object containing the 2D array as a first element (line 8) and its dimensions—the number of rows and columns—as third and fourth elements. Dimensions that we obtain are a sub-list with the equivalent C function of the μLisp `cddr` one (line 9). The first and second elements of this sub-list contain the number of rows and columns (lines 10-11). Next, we create a C language 2D array called `A` (line 13) with the same dimensions as the 2D μLisp array object. Then, in lines 15-21 we iterate over all elements in both the μLisp and the C array to copy all values from the former to the latter. This will allow us to do all the required matrix multiplication computations in C. For the matrix multiplication we will use a readily available Arduino library [5].

In lines 23-36, we perform a similar procedure to obtain a second μLisp 2D array object from the second argument to the function (line 23), and copy its values to a second C language array called `B`. Next, lines 38-39 create a third array named `R` and call the `Matrix.Multiply()` C function to find the product "A*B" and store the result in `R`. Finally, in lines 44-51 we copy the individual elements from `R` to the `resarray` μLisp array object using pointers. As a last step, we return `resarray` to the μLisp environment in the form of another 2D μLisp array.

Matrix multiplication is only possible if the number of columns in the first matrix is equal to the number of rows in the second matrix. That condition isn't checked here, so the user must guarantee the two matrices have the correct dimensions.

FORWARD KINEMATICS IN μLISP

Figure 3 shows the no-frills 3DOF mini robotic arm we used to develop our servomotor control μLisp extension in Part 2. We will use it again here to test the forward kinematics code. Figure 2 and Table 1 show its DH parameters. With that information at hand, let's see how we implement forward kinematics in μLisp. Remember that the focus here is μLisp itself, and the development of extensions to the dialect. Don't worry much about the complexities of robotic kinematics if you are not familiar with it. I really hope, though, that this article fires your interest for further research on the topic.

Listing 3 presents the basic µLisp functions to do forward kinematics. Lines 1-2 define the four parameters from Table 1. These are constants, because they depend solely on the robotic arm's mechanical structure and construction. So they will never change under the normal operation. α_1 is defined in degrees in line 2, but it is later converted to radians (line 8) for all further calculations.

Next, lines 10-27 define the `make-dh-matrix` function that just implements Equation 1 defined previously. It creates a 4 x 4 matrix as a 2D µLisp array (line 11). Then, it populates it with element values computed using Equation 1, which depend on the DH parameters from Table 1. For that, 12 default element values that are "0" are changed using array referencing. The other four remain "0," as per Equation 1.

Next, the function `calc-fkin` from lines 29-38 computes the homogeneous transformations for all coordinate frames in Figure 2 from {1} to {4}. First, it creates the HT transformation matrices T12, T23, and T34 by passing relevant arguments to the `make-dh-matrix` function (lines 31-33). Next, it computes the HT transformations "T13 = T12 * T23" and "T14 = T13 * T34" (lines 34-35). Finally, it returns "T12," "T13," and "T14" as a list (lines 37-38). These matrices contain the pose of frames {2}, {3}, and {4} with respect to frame {1}, which is all we need to locate all robotic arm links in 3D space. T14 contains the 3D coordinates of the end effector with respect to frame {1}.

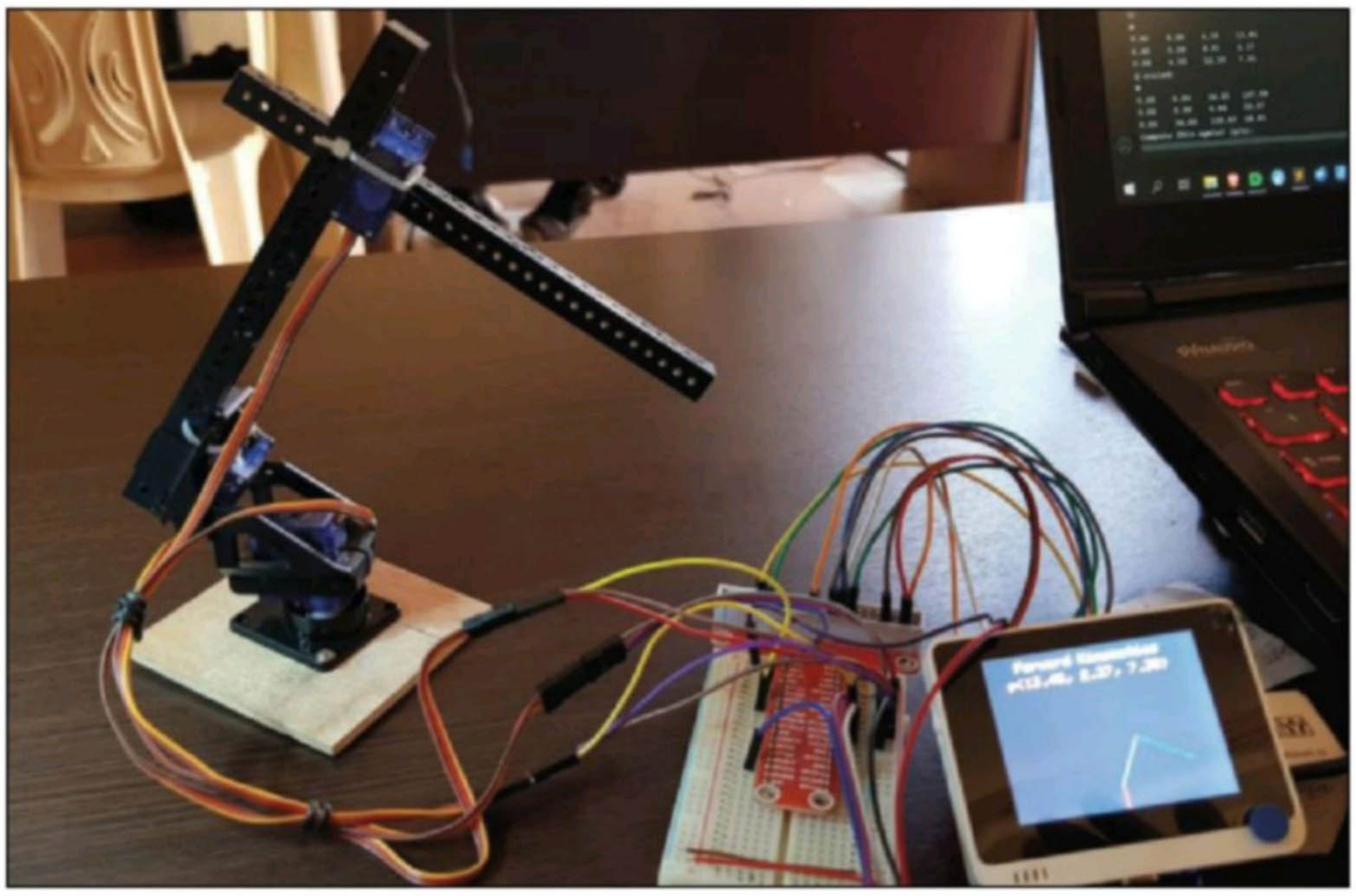
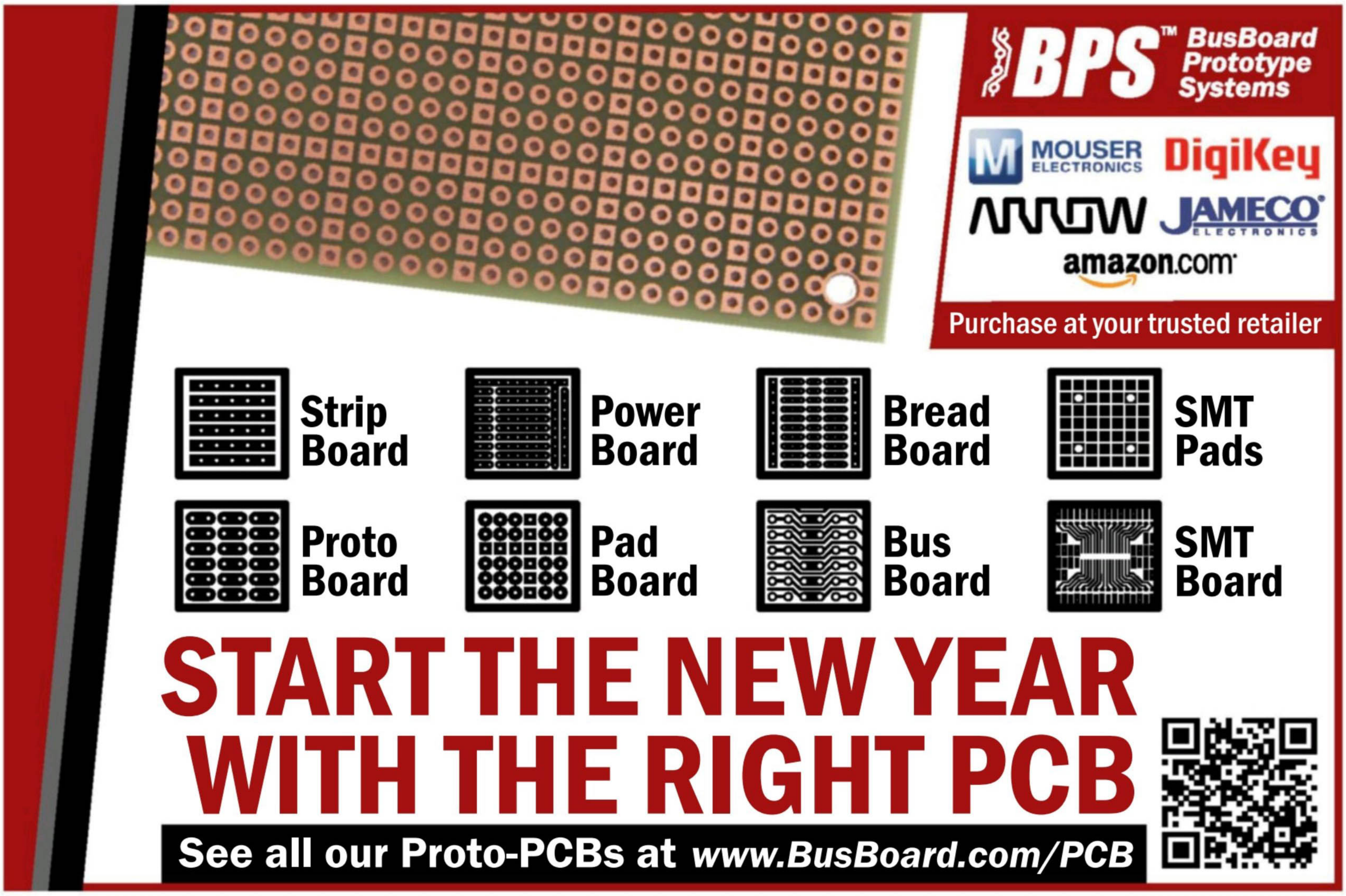


FIGURE 3
The mini robotic arm prototype.



BPS™ BusBoard Prototype Systems

MOUSER ELECTRONICS
DigiKey
ARROW
JAMECO ELECTRONICS
amazon.com

Purchase at your trusted retailer

Strip Board
Power Board
Bread Board
SMT Pads
Proto Board
Pad Board
Bus Board
SMT Board

START THE NEW YEAR WITH THE RIGHT PCB

See all our Proto-PCBs at www.BusBoard.com/PCB

```
1 (defvar d1 4.5) (defvar a2 9.3)
2 (defvar a3 10.4) (defvar alpha1deg 90)
3 (defvar pi 3.141592653589793)
4
5 (defun deg-to-rad (angle)
6  (* angle (/ pi 180)))
7
8 (defvar alpha1 (deg-to-rad alpha1deg))
9
10 (defun make-dh-matrix (theta d a alpha)
11  (let ((m (make-array ‘(4 4) :initial-element 0)) (r))
12    (setf (aref m 0 0) (cos theta) )
13    (setf (aref m 0 1) (-(* (sin theta) (cos alpha))) )
14    (setf (aref m 0 2) (* (sin theta) (sin alpha)) )
15    (setf (aref m 0 3) (* a (cos theta)) )
16
17    (setf (aref m 1 0) (sin theta) )
18    (setf (aref m 1 1) (* (cos theta) (cos alpha))  )
19    (setf (aref m 1 2) (-(* (cos theta) (sin alpha))) )
20    (setf (aref m 1 3) (* a (sin theta)) )
21
22    (setf (aref m 2 1) (sin alpha)  )
23    (setf (aref m 2 2) (cos alpha) )
24    (setf (aref m 2 3) d )
25
26    (setf (aref m 3 3) 1 )
27    m))
28
29 (defun calc-fkin (theta1 theta2 theta3)
30  (let ((t12) (t23) (t34) (t13) (t14) (fkin-result))
31    (setq t12 (make-dh-matrix theta1 d1 0 alpha1))
32    (setq t23 (make-dh-matrix theta2 0 a2 0))
33    (setq t34 (make-dh-matrix theta3 0 a3 0))
34    (setq t13 (mat-mult t12 t23))
35    (setq t14 (mat-mult t13 t34))
36
37    (setq fkin-result (list t12 t13 t14))
38    fkin-result))
39
40 (defun get-link-coords (fkin-result)
41  (let ((Q (make-array ‘(3 4) :initial-element 0)))
42    (setf (aref Q 0 1) (aref (first fkin-result) 0 3))
43    (setf (aref Q 0 2) (aref (second fkin-result) 0 3))
44    (setf (aref Q 0 3) (aref (third fkin-result) 0 3))
45
46    (setf (aref Q 1 1) (aref (first fkin-result) 1 3))
47    (setf (aref Q 1 2) (aref (second fkin-result) 1 3))
48    (setf (aref Q 1 3) (aref (third fkin-result) 1 3))
49
50    (setf (aref Q 2 1) (aref (first fkin-result) 2 3))
51    (setf (aref Q 2 2) (aref (second fkin-result) 2 3))
52    (setf (aref Q 2 3) (aref (third fkin-result) 2 3))
53    Q))
54
55 (defun run-plot-fkin ()
56  (loop
57    (let ((th1) (th2) (th3) (fkin-result) (Q))
58     (format t “Enter theta1, theta2, theta3:~%”)
59     (setq th1 (deg-to-rad (string-to-float (read-line))))
60     (setq th2 (deg-to-rad (string-to-float (read-line))))
61     (setq th3 (deg-to-rad (string-to-float (read-line))))
62     (setq fkin-result (calc-fkin th1 th2 th3))
63     (format t “fkin-result:~A~%” fkin-result)
64
65     (setq Q (get-link-coords fkin-result))
66     (format t “Q:”) (print-mat Q)
67     Q)
68    (format t “Compute fkin again? (y/n):~%”)
69    (if (= 110 (char-code (char (read-line) 0)))
70     (return))
71    ))
72
73 (run-plot-fkin)
```

LISTING 3
Forward kinematics µLisp code.

The `get-link-coords` function from lines 40-53 takes the aforementioned `fkin-result` list containing T12, T13, and T14, and extracts from each one of them the elements that constitute the x, y, and z coordinates for all link-ends. **Figure 4** shows which elements those are. These coordinates are then returned in the "Q" matrix.

Finally we have the `run-plot-fkin` function that serves as the entry point for the whole application. In this reduced version of the code, it just computes and prints to the REPL the matrices "fkin-result" and "Q" containing the x, y, and z coordinates of all link-ends. This function runs as an endless loop (Line 56) that breaks when the user types "n," after answering the choice given in line 68. The breaking condition is detected as follows. Line 69 reads a character from the REPL: `(char (read-line) 0)`, then it gets its character code: `(char-code (...))` and compares it with the ASCII code of character "n", which is 110: `(if (= 110 (...)) (return))` to exit from the infinite loop.

The complete code, available for download from the *Circuit Cellar* Code and Files webpage, includes functions to plot a 2D graph in the x-z planar coordinate system of frame {1}. **Figure 5** shows the plot on the Wio Terminal's LCD display. A second plot in x-y can also be generated using a similar approach.

3D perspective plots are also possible applying further homogeneous transformations specifically for generating the 3D perspective, but that's even further out of this article's scope. The full source code also includes the functions written in Part 2 to drive the robotic arm servomotors.

TESTING

If you decide to build your own robotic arm, please take into account that depending on how you position the servomotors, joint rotation directions can get inverted. It is advisable to check this first, before fixing the servos on the structure. Otherwise, you must correct it by modifying the code accordingly. The only joint disalignment taken into account in the code is "d1" along frame {1}'s z-axis between frames {1} and {2}. Your structure must not have other joint disalignments—that is, frame displacements—along the z-axis. Otherwise, the code must also be modified to take this into account.

Robotic arms have an associated property called "workspace." The workspace is the combination of all possible 3D positions the end effector can reach. All those 3D points have corresponding sets of joint angle combinations. As you might imagine, there are, of course, positions that are not possible for a given robotic arm, because of its geometric constraints. For instance, the end effector can never reach points beyond the total length of the extender arm in all feasible directions, nor can it reach positions that are under its base (if the arm is on top of a table, for example), and so on.

The code presented in this article doesn't account for these cases, so when testing it, you must consider this. There's also an angle range constraint for typical servomotors such as the ones used here, which is normally 0-180°. For the prototype showcased in this article, I chose a particular range mapping from the logical to the physical range constraint, which is shown in **Figure 6**.

CONCLUSION

There are both advantages and disadvantages of using Lisp languages in general, and μLisp in particular. All Lisps generally are very useful for rapid-prototyping, due to their interactive nature. Even Lisps that are compiled act as interpreted languages, providing the advantages associated with these—for instance, the interactive REPL. An interesting example of this is that you can change a function and compile just that function, without stopping the execution of the rest of the program that calls it.

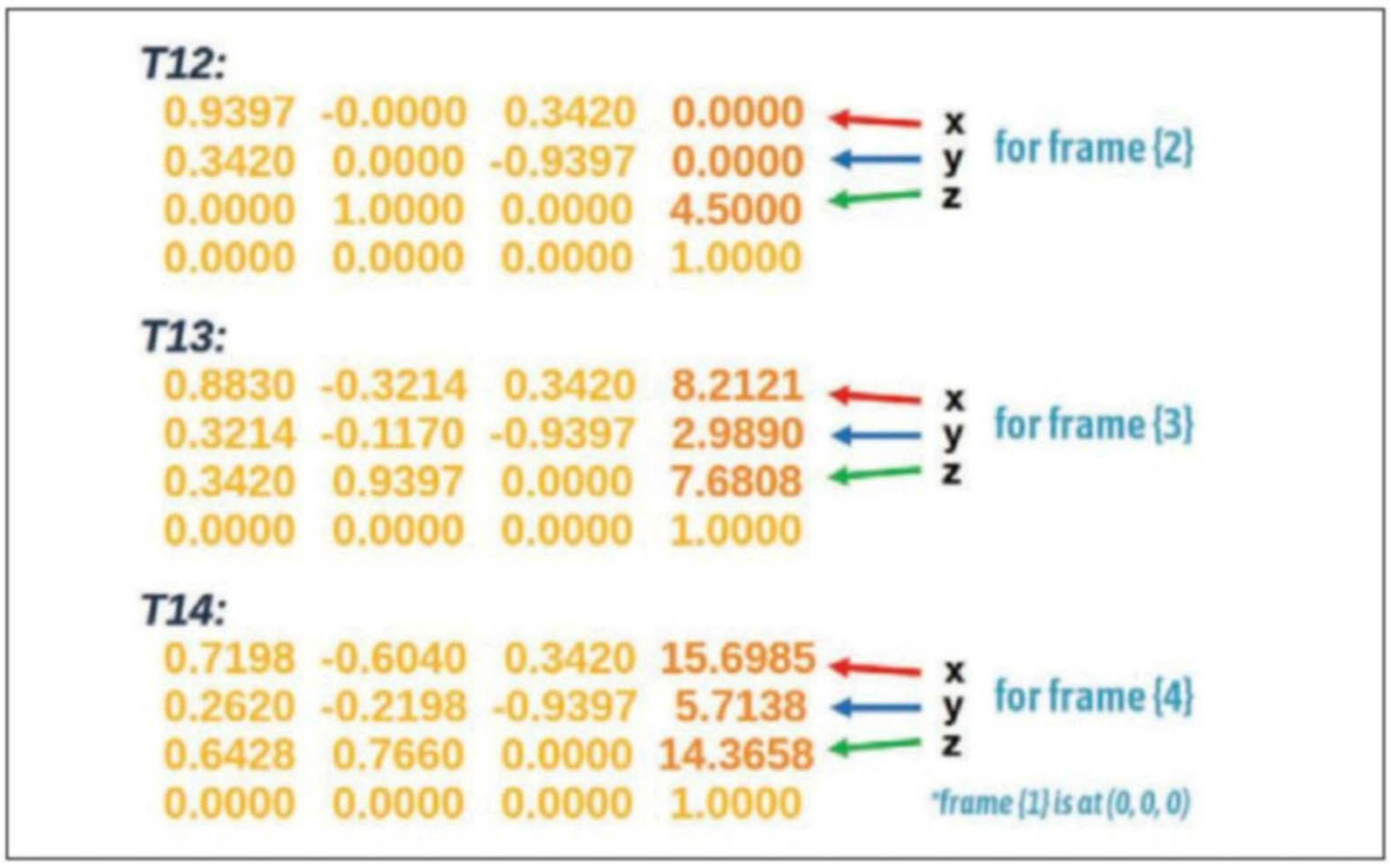
μLisp, itself, is just an interpreted dialect; nevertheless, it carries the philosophy of its bigger brothers. Being a declarative, functional language, it lets you express solutions that are simpler and more intuitive than using imperative programming such as with C language.

In contrast, the main disadvantage of μLisp is the memory overhead imposed by the interpreter and the garbage collector, which make the code execution slower in comparison with C. As a result, μLisp probably is not feasible for applications where timing is critical.

If you are already familiar with MCUs and want to introduce yourself to Lisp programming, I think μLisp combines both interests nicely. It supports a wide variety of MCU platforms, and the website has great documentation and also a forum where you can ask questions.

Some functions you might want to use may not be available by default; however they can be added as extensions reasonably easily. For example, in this series we implemented extensions such as "micros," "pulse-in," "servo-create," "servo-write," "mat-mult," "mat-print," and "string-to-float" without much hassle. In the end, this was also a good way of getting to know how the μLisp interpreter is implemented in C.

I hope you enjoyed the series and that it sparked your interest in Lisp, which is a most powerful language. Thanks for your attention—until the next time!



T12:

0.9397	-0.0000	0.3420	0.0000	x
0.3420	0.0000	-0.9397	0.0000	y
0.0000	1.0000	0.0000	4.5000	z
0.0000	0.0000	0.0000	1.0000	

for frame {2}

T13:

0.8830	-0.3214	0.3420	8.2121	x
0.3214	-0.1170	-0.9397	2.9890	y
0.3420	0.9397	0.0000	7.6808	z
0.0000	0.0000	0.0000	1.0000	

for frame {3}

T14:

0.7198	-0.6040	0.3420	15.6985	x
0.2620	-0.2198	-0.9397	5.7138	y
0.6428	0.7660	0.0000	14.3658	z
0.0000	0.0000	0.0000	1.0000	

for frame {4}

*frame {1} is at (0, 0, 0)

FIGURE 4
Matrices T12, T13, and T14 containing coordinates for all lin- ends.

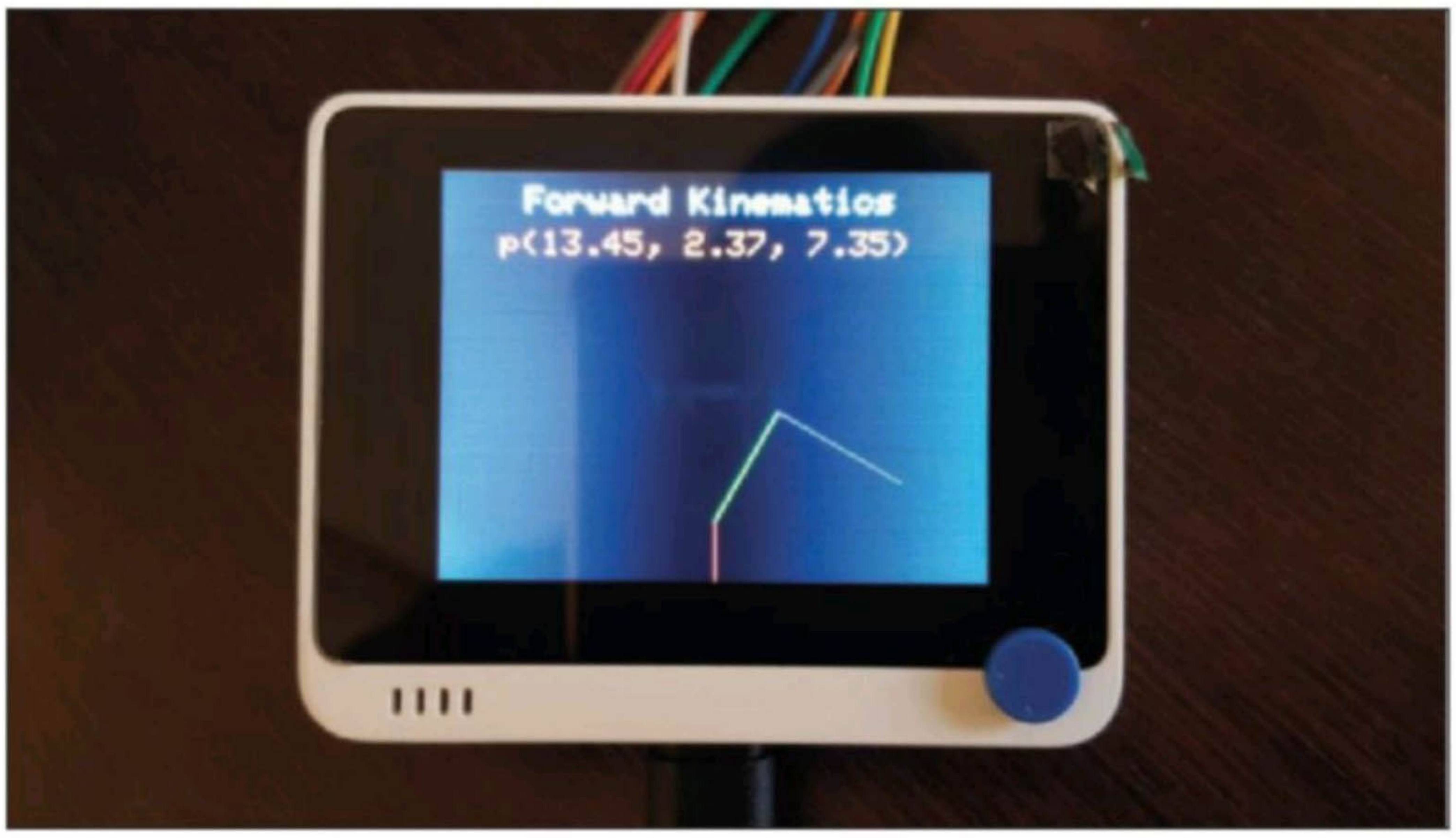


FIGURE 5
Forward kinematics graph in the Wio Terminal LCD.

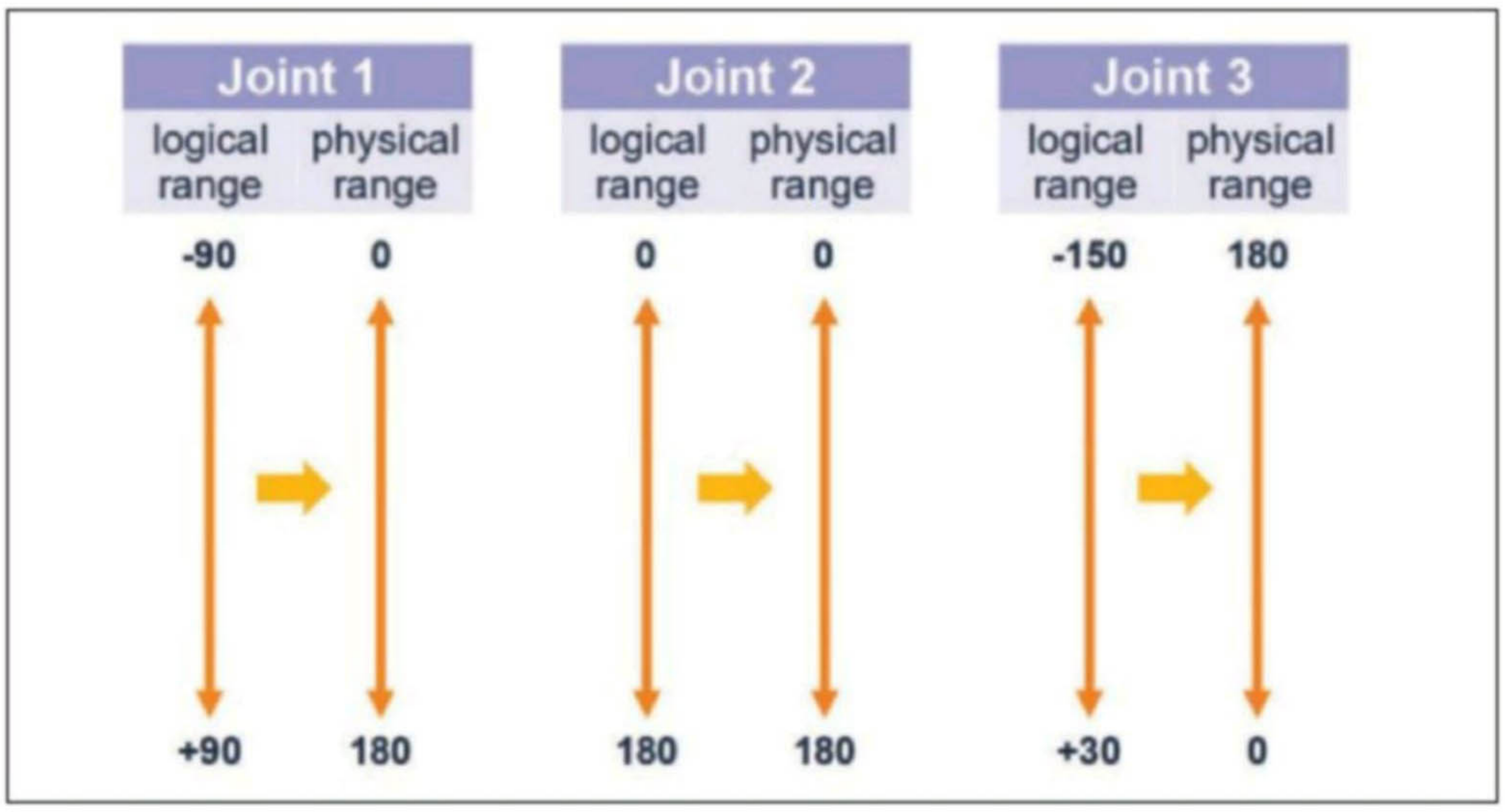


FIGURE 6
Logical to physical angle range mappings.

Additional materials from the author are available at:
www.circuitcellar.com/article-materials
References [1] to [5] as marked in the article can be found there.

Introducing Bluetooth LE Audio

Part 2—Learning the Intricacies of these New Specifications

By
Nick Hunn

Bluetooth Low Energy (LE) Audio introduces a revolutionary set of specifications, following the Bluetooth Core 5.2 release, that redefine audio applications through isochronous channels and a new audio middleware. In this two-part series, Nick provides an overview of the new specifications and discusses new ways to experience and share sound.

This comprehensive framework supports bidirectional audio, broadcast capabilities for shared listening experiences, and utilizes the high-efficiency LC3 codec to enhance audio quality for various applications, including personal and public broadcasting. Here, we continue our overview of the new Bluetooth LE Audio specifications, including Broadcast Isochronous Streams (BISes).

BROADCAST ISOCHRONOUS STREAMS

BISes look much like Connected Isochronous Streams (CISes), but with one exception—there are no acknowledgments. Broadcast audio is a one-way process, where a Broadcast Transmitter (also defined as an Initiator) transmits with no knowledge of whether anything is present to receive its transmissions. It may be received by one device, or by a hundred. In a recent demonstration in the Alice Tully concert hall in New York's Lincoln Center, a single transmitter covered an auditorium of more than 1,000 people.

Although the basic structure of a BIS is similar to a CIS, there are a few subtle differences. Each Broadcast Transmitter is configured for a set number of retransmissions, which are unconditionally sent,as there are no acknowledgments to indicate that they have been received. Accepters can stop listening for them as soon as they have received a valid packet, putting their radio to sleep until the next Isochronous Interval.

Another difference is that while different CISes in a Connected Isochronous Group (CIG) can have different properties, such as different codec configurations which result in different sized packets, every BIS has the same structure. Different BISes can transport different sized packets, but smaller packets will just waste space. If you want to support two streams, such as one sampled at 24kHz and another at 48kHz, it is generally more efficient to use two different BIGs, each optimized for their particular stream.

Most people are surprised by the range of a broadcast. Normally, the range of a Bluetooth connection is governed by the device with the weaker transmitter, as that determines the link budget, and hence the range. For audio, that is invariably the earbud. Not only does it have a weaker transmitter and smaller antenna, but one earbud is likely to have its signal attenuated by your head. It's not unusual to see a link budget asymmetry of 50dB or more, which is why the range is typically around 5m to 10m. With no weak acknowledgments to consider, Broadcast Transmitters can easily cover 100m, or more. That makes them eminently suitable for public broadcast, as well as personal applications.

In broadcast applications, the Broadcast Transmitter does not normally pair with any Acceptor. It transmits unconditionally, hoping that something is listening. That means that there is no negotiation about the capabilities of the Acceptors that want to listen to it. As many of the features of the Bluetooth LE Audio specification are optional, there is a risk if

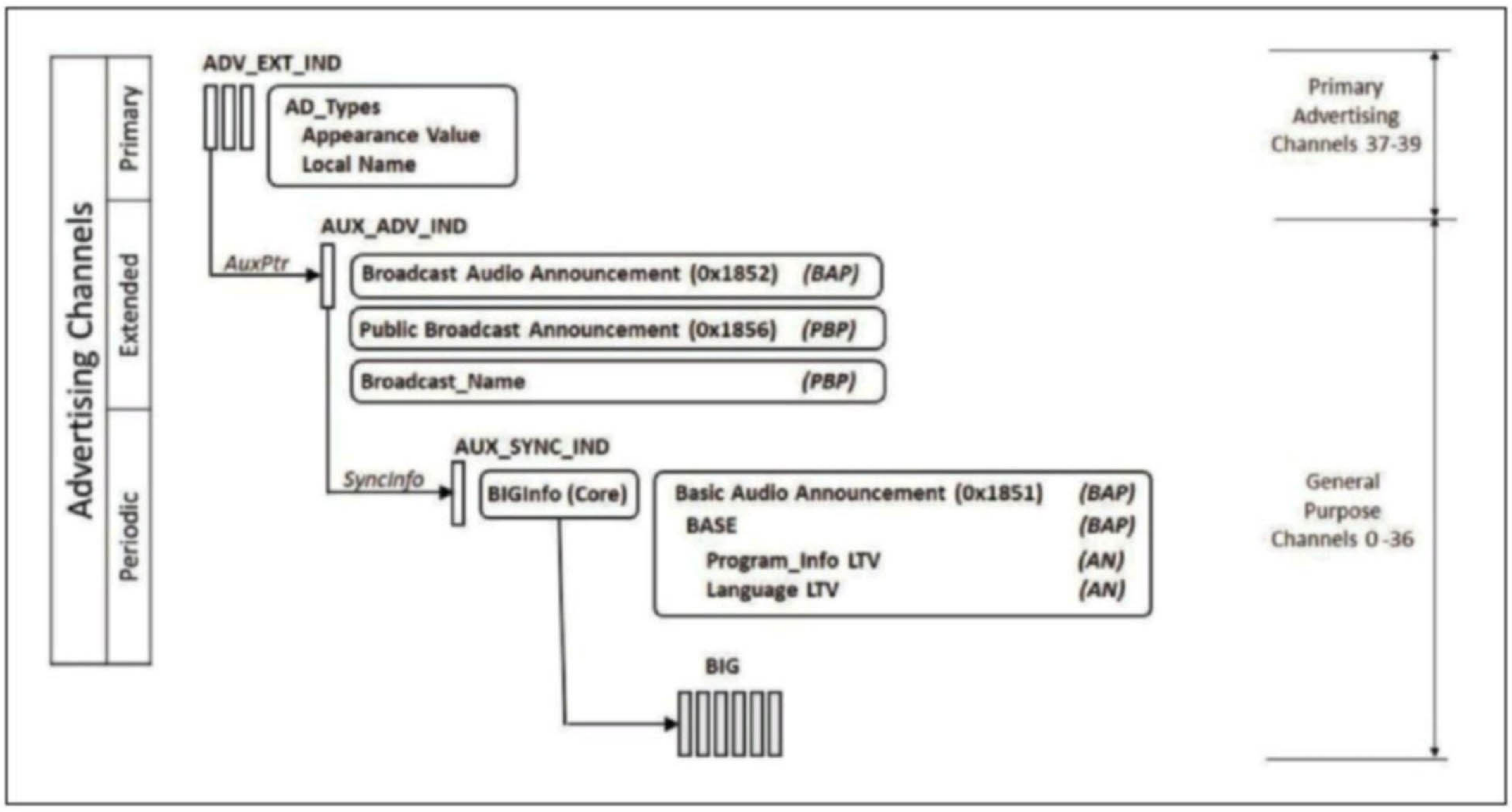

Advertising Channels
Primary
Extended
Periodic
ADV_EXT_IND
AD_Types
Appearance Value
Local Name
AuxPtr
AUX_ADV_IND
Broadcast Audio Announcement (0x1852) (BAP)
Public Broadcast Announcement (0x1856) (PBP)
Broadcast_Name (PBP)
AUX_SYNC_IND
SyncInfo
BIGInfo (Core)
Basic Audio Announcement (0x1851) (BAP)
BASE (BAP)
Program_Info LTV (AN)
Language LTV (AN)
BIG
Primary Advertising Channels 37-39
General Purpose Channels 0-36

FIGURE 5
Extended advertising for Broadcast Streams.

Broadcast Transmitters use optional features, such as higher quality codec settings, that some Acceptors may not be able to decode them. For this reason, most Broadcast Transmitters restrict their configurations to the minimum mandated settings, which every Acceptor has to support. For a codec, that a sampling rate is 24Hz using LC3, with a Presentation Delay of 40ms.

The other consequence of the lack of any connection is that Acceptors have to discover the existence of a Broadcast Source by themselves. They do this using the extended advertising features, which were adopted in the Core v5.1. Extended advertising allows more information to be transmitted than what will fit into the three dedicated advertising channels of Bluetooth LE.

Figure 5 shows how this works. A normal Primary Advertisement includes a "AuxPtr" flag to indicate that Extended Advertisements exist. For a Broadcast Transmitter, these Extended Advertisements include a Broadcast Audio Announcement to inform devices that it is broadcasting audio. It also includes its name and high-level information about its configuration as defined by the Public Broadcast Profile (PBP). A Broadcast Receiver can use this information to determine whether it is a broadcast of interest to its user.

The Extended Advertisement also includes a link to a Periodic Advertisement, which contains two important structures. These are the Basic Audio Announcement, which contains detailed information about the content of the broadcast. A user application will use this information to determine whether to "connect" to the stream based on its content. Once that decision is made, it will retrieve the BIGInfo structure, which contains the low-level information on where to find the BIG, including its hopping pattern, its timing and the structure of the BIG, and its constituent BISes. With that knowledge, we can move up the stack and see how to apply the elements of the Generic Audio Framework (GAF).

CONFIGURING CISES AND BISES

Stream control and configuration takes us into the four main specifications of the GAF.

- The Basic Audio Profile (BAP) is used to configure both CISes and BISes.
- The Audio Stream Configuration Service to set up CISes on Acceptors.
- The Published Audio Capabilities Service (PACS) exposes the features that an Acceptor supports for both broadcast and unicast streams.
- The Broadcast Audio Stream Service (BASS) specifies ways for an Acceptor to find and synchronize to a Broadcast, with the help of a third device.

Taken together, these four specifications are referred to as BAPS, and are the fundamental elements of the GAF.

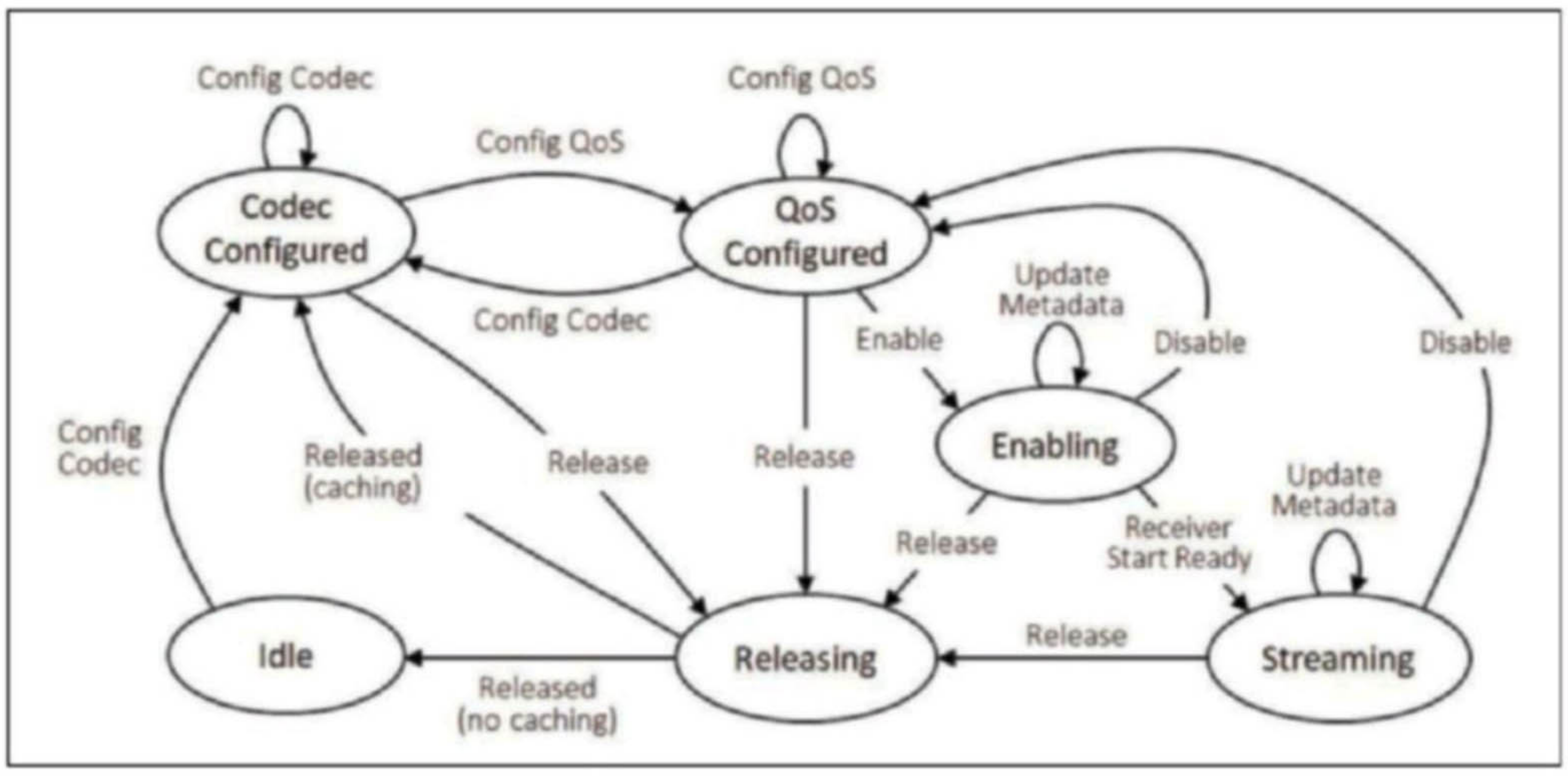
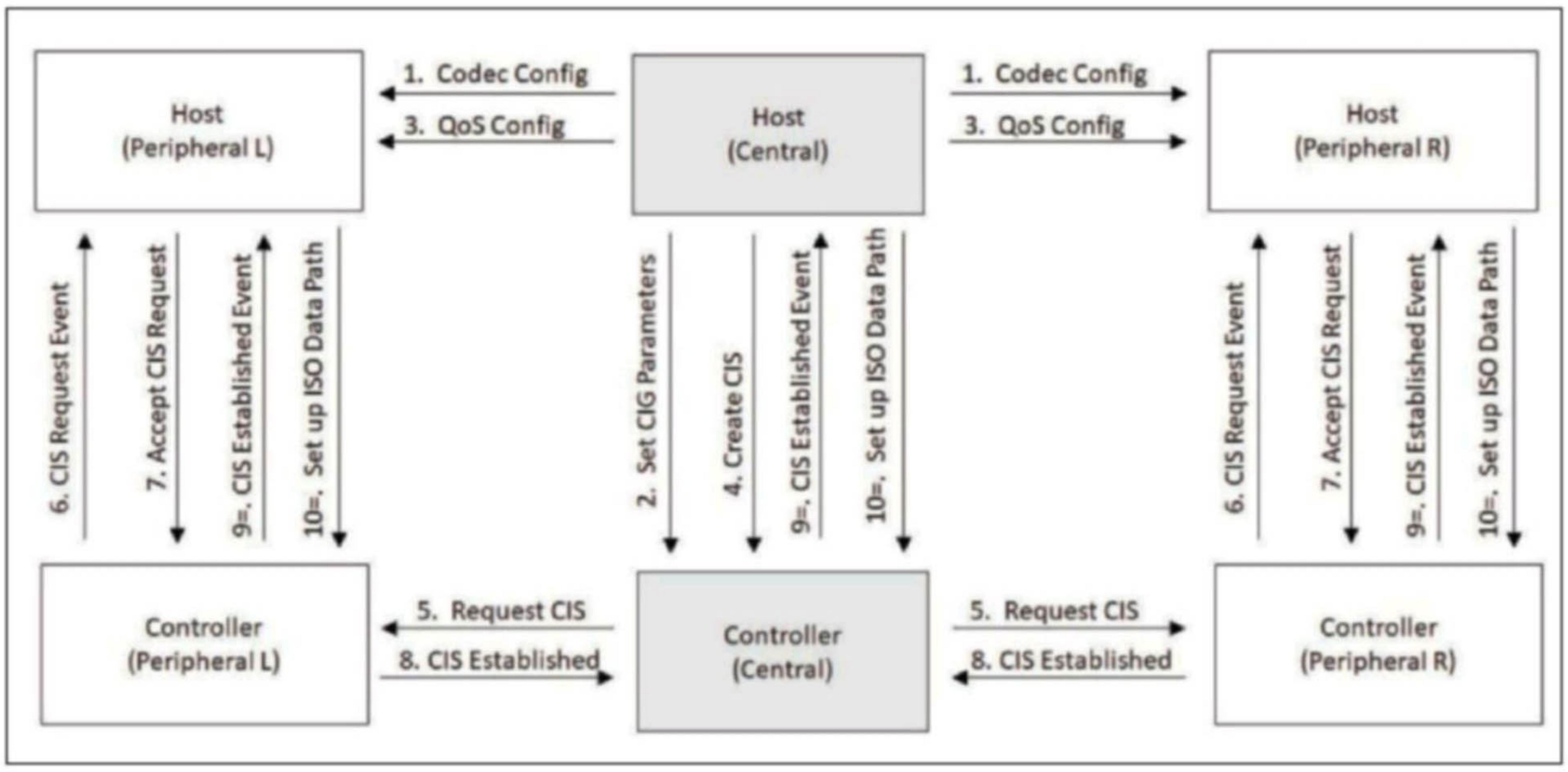

Config Codec
Config QoS
Config QoS
Codec Configured
QoS Configured
Config Codec
Update Metadata
Enable
Disable
Disable
Config Codec
Released (caching)
Release
Release
Enabling
Update Metadata
Release
Receiver Start Ready
Idle
Releasing
Release
Streaming
Released (no caching)

FIGURE 6
The state machine for a Sink ASE.

FIGURE 7
The configuration process for a CIG with two ASEs.


Host (Peripheral L)
1. Codec Config
3. QoS Config
Host (Central)
1. Codec Config
3. QoS Config
Host (Peripheral R)
6. CIS Request Event
7. Accept CIS Request
9=. CIS Established Event
10=. Set up ISO Data Path
2. Set CIG Parameters
4. Create CIS
9=. CIS Established Event
10=. Set up ISO Data Path
6. CIS Request Event
7. Accept CIS Request
9=. CIS Established Event
10=. Set up ISO Data Path
Controller (Peripheral L)
5. Request CIS
8. CIS Established
Controller (Central)
5. Request CIS
8. CIS Established
Controller (Peripheral R)

Starting with unicast (i.e., CISes and CIGs), BAP defines Audio Stream Endpoints (ASEs) which exist on an Acceptor for every CIS. Each ASE defines the entire configuration of that stream. For bidirectional streams, one ASE exists on the Acceptor for each direction—a Sink ASE, which receives audio data from the Initiator and a Source ASE, which returns audio captured by the Initiator. Each are configured using a state machine defined in BAP.

Figure 6 shows the state machine for a Sink ASE. (The Source ASE state machine has one additional state to ensure a clean termination of audio captured at the Acceptor, but the principle of configuration is essentially the same).

The state machines for ASEs exist on the Acceptor. An Initiator configures the ASE by sending Control Point operations to the appropriate ASE characteristic, which moves it through the states until it is streaming.

All CISes require the Initiator and Acceptor to be paired in advance. At this point the Initiator can read the characteristics in the Acceptor's Published Audio Capabilities Service to help it with its configuration choices. When an Initiator wants to set up a Sink ASE to transport audio to an Acceptor it starts by sending an opcode to move the ASE to its Codec Configured state, specifying the codec configuration it wants to use, and preferences for items such as the maximum transport latency. The ASE will respond with its corresponding values and provide preferred options for its Quality of Service (QoS) configuration. These include preferred Presentation Delay and the number of retransmissions.

Based on these responses, the Initiator should move all of the ASEs it needs on all of its Acceptors to the QoS configured state, ensuring that the configuration is consistent across them all. It then writes these to the Core controller, specifying the entire structure of the CIG and its constituent CISes, using HCI commands, which have been introduced to allow the Host to configure Isochronous Channels. If the Host is happy with these, it creates the CIG by establishing at least one of the CISes by moving it to the Enabled state.

Figure 7 shows the sequence of events for an Initiator that is streaming to a pair of stereo earbuds. To move a CIS to its final streaming state, each device needs to set up the data path for each ASE, telling it where the codec implementation for that ASE is situated. The specification allows the codec to reside in either the host or the controller. At that point, streaming can start. After the first CIS in the CIG has been enabled, other CISes can be turned on or off to suit the application. This allows CISes to be re-purposed for other applications. However, the configuration of the ASE cannot be changed without disabling the entire CIG and reconfiguring it. When an ASE is terminated, the QOS configuration state is normally cached, allowing simpler and faster configuration the next time it is used.

CONFIGURING A BROADCAST TRANSMITTER

Setting up a Broadcast Transmitter is much easier, as it doesn't need to collect information from any of its potential Acceptors. You just decide how you want to configure it, set it up, and turn it on. There is a very simple state

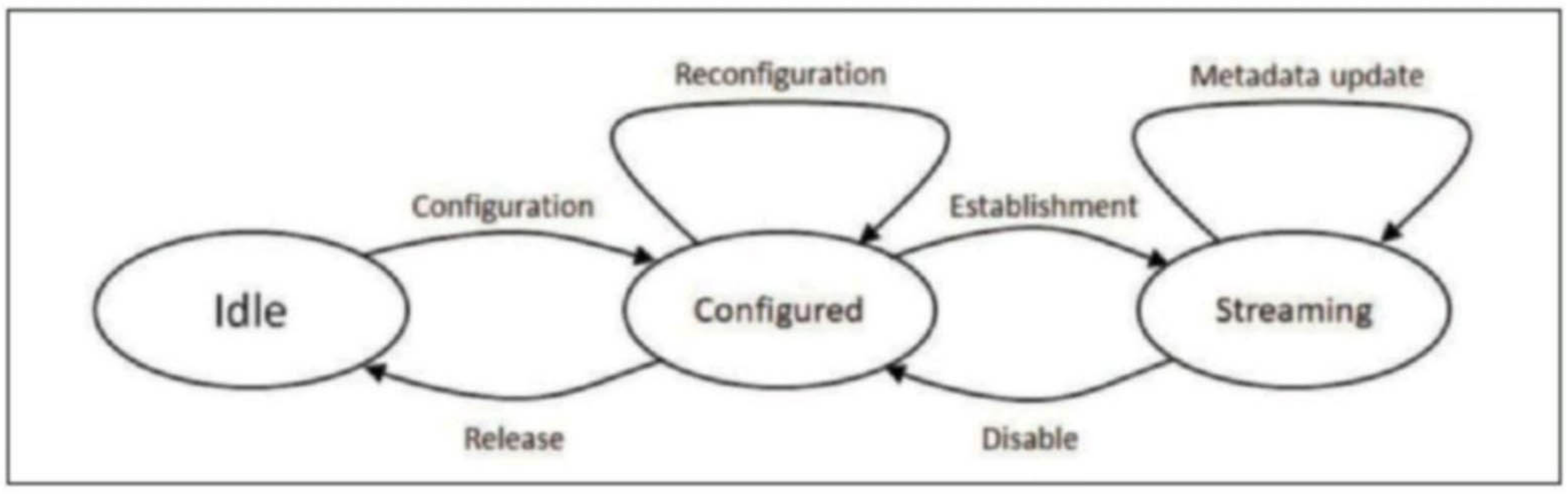

Reconfiguration
Metadata update
Configuration
Establishment
Idle
Configured
Streaming
Release
Disable

FIGURE 8
The state machine for a Broadcast Transmitter.

machine, which in this case resides on the Initiator (**Figure 8**).

All that a host application needs to do is define the configuration, then use an HCI command to establish the BIG and start streaming.

A Broadcast Transmitter also needs to supply information for the advertisements, which Acceptors will use to find the broadcast streams and discover what is in them. Most of this information is included within the Broadcast Audio Stream Endpoint structure that is included in the Basic Audio Announcement within the Periodic Advertisements. As well as identifying the different BISes within a BIG, it also contains metadata that provides information about the content. Some of this is in human readable form to help inform users of what the transmitter is providing.

Figure 9 shows an example of the BASE structure for a BIG, which contains two stereo streams divided into two pairs of BISes—one in English and the other in Spanish. The information in the BASE allows an Acceptor that is scanning for these advertisements to understand the configuration of the BIG, the relative positions of each BIS within it, and human readable metadata elements that can help a user identify and choose the content.

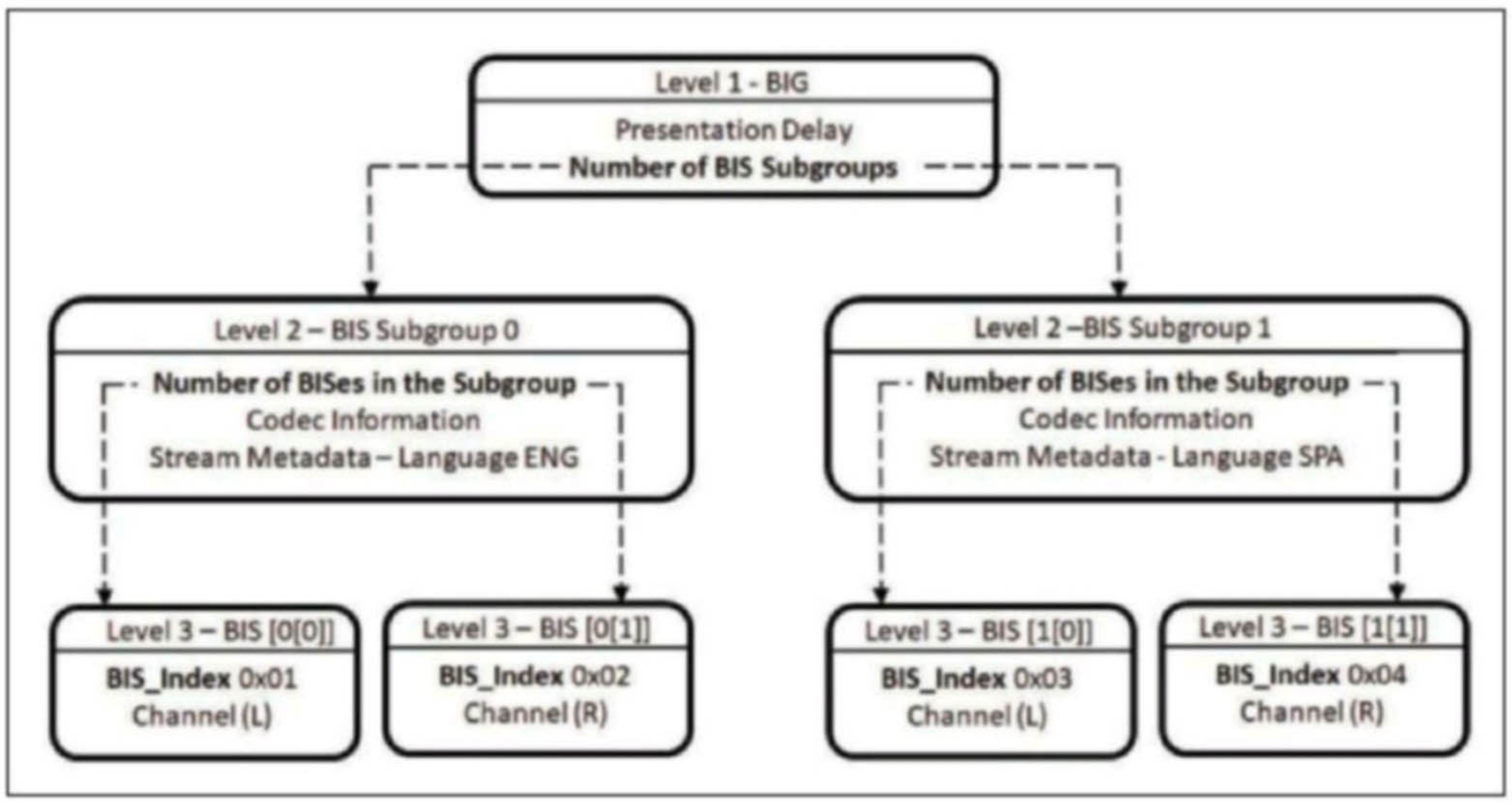
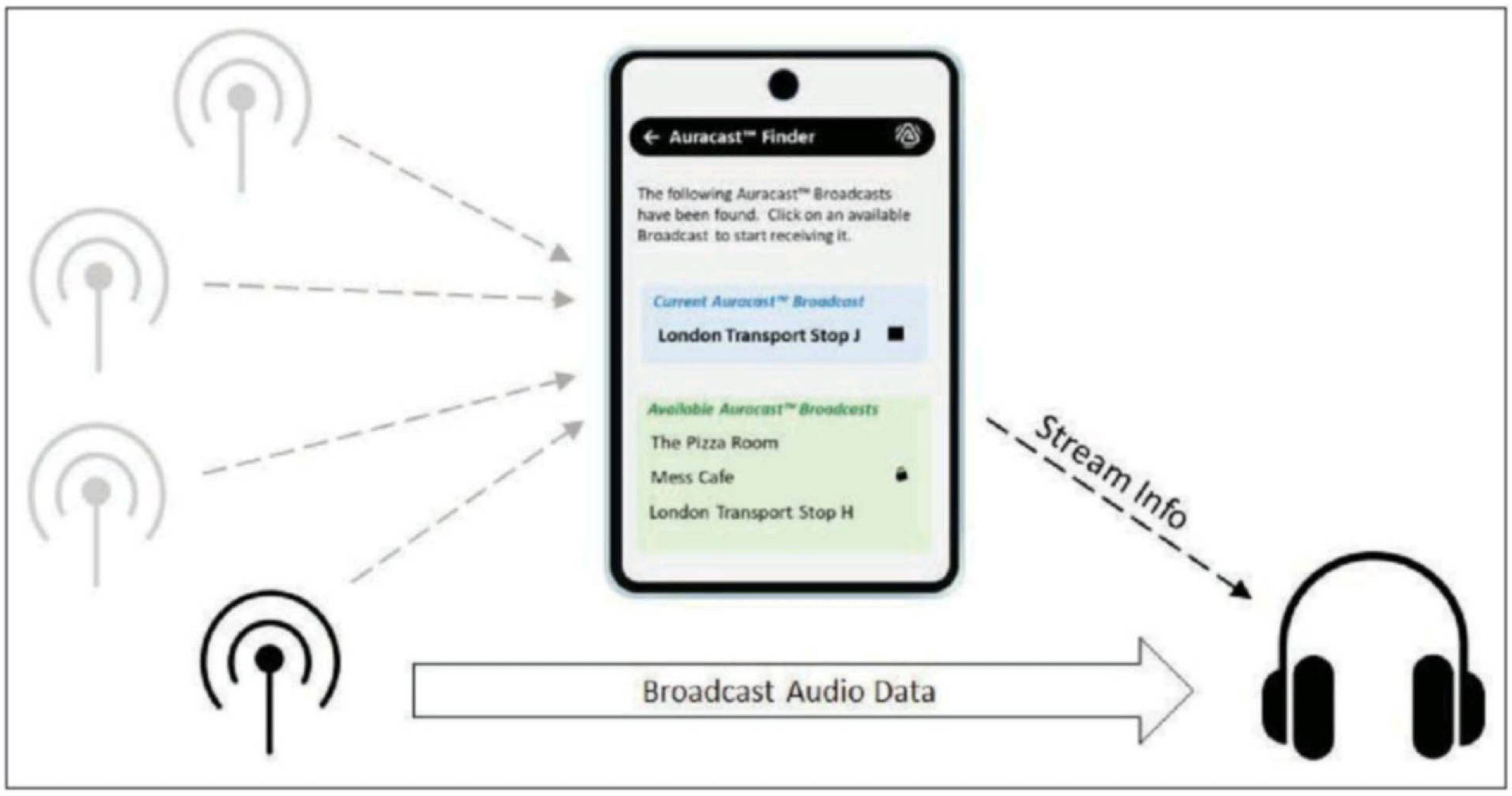

Level 1 - BIG
Presentation Delay
Number of BIS Subgroups
Level 2 – BIS Subgroup 0
Number of BISes in the Subgroup
Codec Information
Stream Metadata – Language ENG
Level 2 –BIS Subgroup 1
Number of BISes in the Subgroup
Codec Information
Stream Metadata - Language SPA
Level 3 – BIS [0[0]]
BIS_Index 0x01
Channel (L)
Level 3 – BIS [0[1]]
BIS_Index 0x02
Channel (R)
Level 3 – BIS [1[0]]
BIS_Index 0x03
Channel (L)
Level 3 – BIS [1[1]]
BIS_Index 0x04
Channel (R)

FIGURE 9
BASE structure for a dual-language stereo BIG.

The user identification aspect is important, as there may be multiple Broadcast Transmitters within range of an Acceptor. Unlike telecoils—the precursor of Bluetooth LE Audio broadcasts—Bluetooth can pass through walls, so users may have multiple different Broadcast Transmitters from which to choose. Every Acceptor capable of receiving broadcasts must be able to scan for these advertisements, but it's difficult to implement useful user interfaces on earbuds and hearing aids. To address this, the Bluetooth LE Audio specifications allow an Acceptor to delegate the scanning function to another device, such as a phone, which can provide a more comprehensive user interface, allowing a user to select which broadcast they want to receive. The phone does not receive the audio stream. It just scans for the advertisements, provides a selection app for the user and then instructs the Acceptor to synchronize to the selected stream using a procedure called Periodic Advertising Synchronization Transfer (PAST). **Figure 10** shows the process and an example of a phone app displaying four Broadcast Transmitters it has found, with the user selecting one of them.

Devices that take on the role of scanning are called Broadcast Assistants and can be implemented on any device that is capable of scanning. As well as phones, they can be implemented as apps on smart watches, built into battery boxes, or made as stand-alone remote controls. Multiple Broadcast Assistants are allowed, operating on a first come, first served basis. They serve a useful secondary purpose, as the task of scanning is relatively power hungry. Delegating it to a device with a larger battery is advantageous for devices with small batteries (e.g., earbuds and hearing aids).


← Auracast™ Finder
Stream Info
Broadcast Audio Data

The following Auracast™ Broadcasts have been found. Click on an available Broadcast to start receiving it.

Current Auracast™ Broadcast
London Transport Stop J

Available Auracast™ Broadcasts
The Pizza Room
Mess Cafe
London Transport Stop H

FIGURE 10
Using a Broadcast Assistant to acquire an Auracast stream.

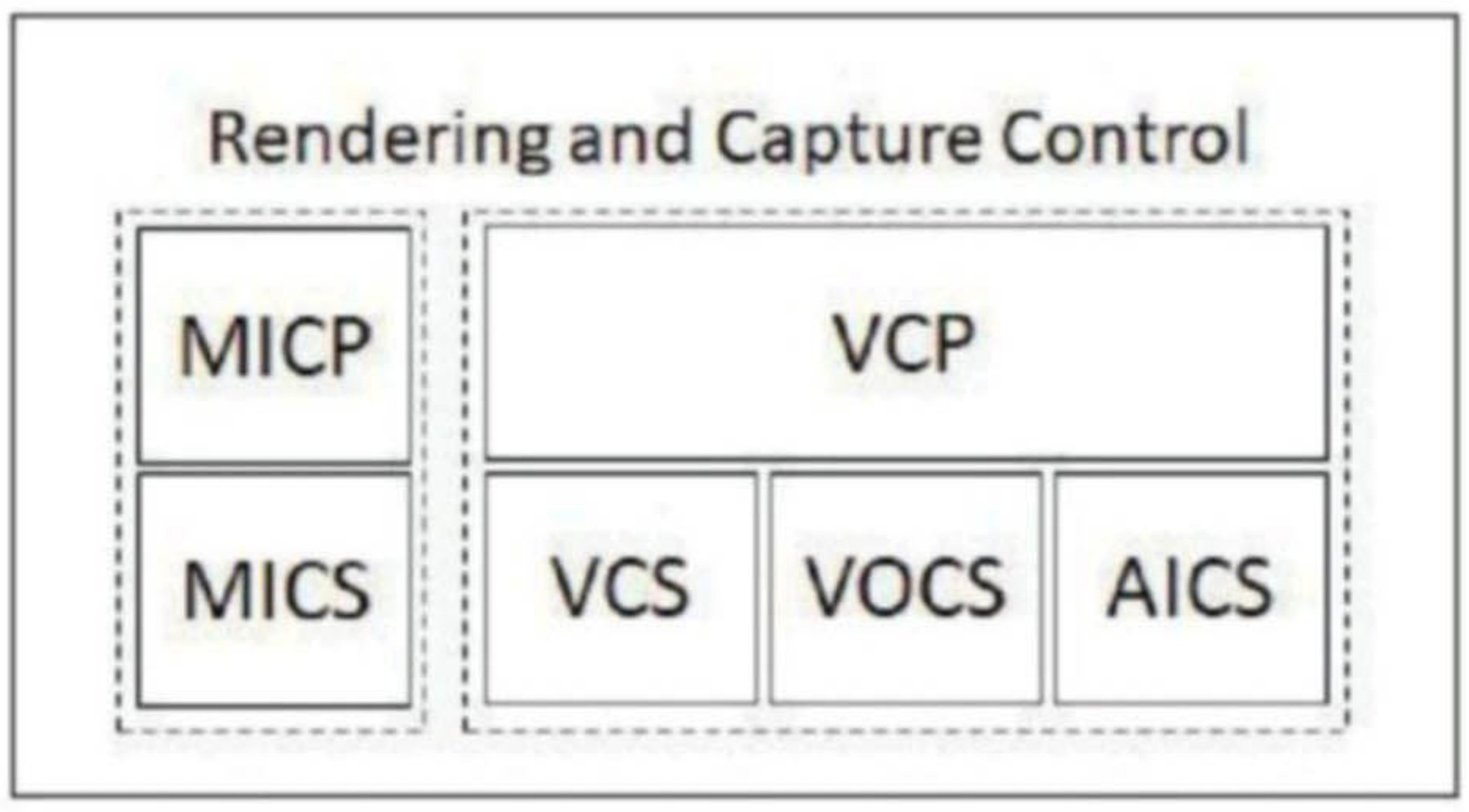


FIGURE 11
Rendering and capture control.

Because they may carry audio signals from a personal device or private conversations, broadcast audio streams can be encrypted, using a password to limit access to authenticated users.

OTHER GAF SPECIFICATIONS

It's possible to design audio applications that only use the BAPS specifications, which were designed to ensure a base level of features to ensure interoperability. However, once you move away from a single audio stream, or want to include more advanced features, you need two other pairs of specifications.

The first of these are the Coordinated Set Identification Profile and Service (CSIP and CSIS). Coordination is a new concept that identifies two or more Acceptors, which work together as if they were a single device. The most common example is left and right earbuds or speakers. When your phone connects to them, it needs to know that it has to find both of them and stream the appropriate audio stream to each one. CSIP and CSIS define how coordination is achieved, with a shared key to identify the members. It also allows coordinated devices to be locked so that a single Initiator can act preferentially upon them, preventing a second Initiator trying to make changes to just one of the pair.

The Common Audio Profile (CAP) builds upon BAPS, pulling in CSIS and CSIPs to define common procedures for establishing, modifying and terminating isochronous streams to any number of Acceptors. The Common Audio Service (CAS) is a minimal service for CAP, which exposes supported features. CAP also provides integration of the control services which operate on Bluetooth LE Audio devices. All of the Top Level Profiles build upon CAP.

TELEPHONY AND MEDIA CONTROL

In the Advanced Audio Distribution Profile (A2DP) and the Hands Free Profile (HFP), application control was included in the main profile. With Bluetooth LE Audio telephony control and media control have their own sets of Profiles and Services—the Call Control Profile (CCP) and Telephony Bearer Service (TBS) for calls, and the Media Control Profile (MCP) and Service (MCS) for music streaming. Both contain a generic state machine that resides on the Initiator, providing independent control of the state of calls or media.

CCP and TBS are designed to be generic, working for any type of call, whether that's cellular, Voice over IP (VoIP), or anything else, getting over the restrictions that were present with HFP. TBS allows multiple instances, so that different call applications can be controlled directly by a Client, or a generic instance, where individual telephony applications are managed by the host.

MCS and MCP are similarly generic, supporting any audio media application. They include comprehensive features for audio stream manipulation, both in terms of searching for audio content, as well as movement through audio tracks and adjusting the playing speed. All of these are commands sent from a device (e.g., a headset) to the Initiator, which is hosting the media application. This topology means that separate devices (e.g., a smart watch), can be used to control the state machine. Multiple Clients can operate on the Server, so both a watch and an earbud can be used to control audio playback, allowing far more flexible topologies.

VOLUME

Volume control is a surprisingly complicated issue, particularly when an earbud or hearing aid can mix multiple Bluetooth and ambient inputs. To cope with this complexity, there are six different specifications that relate to volume and muting on an Acceptor (**Figure 11**).

In another change to classic audio solutions, the volume level, or more correctly gain, is a state that is held independently on each Acceptor. There is an assumption that Bluetooth LE Audio signals are encoded at a line level, without the volume being adjusted on the transmitted stream. The state of the volume is held by the Volume Control Service (VCS) characteristic. For stereo devices, a separate Volume Offset Control Service (VOCS) allows a Client to add an offset, effectively acting as a balance feature.

To cope with different audio inputs, the Audio Input Control Service (AICS) provides independent volume control and muting on devices that contain multiple audio inputs. All of these services are controlled by a Client implementing the Volume Control Profile. As we've seen above, multiple devices can act as Clients on each Acceptor. To prevent

Additional materials from the author are available at:
www.circuitcellar.com/article-materials

RESOURCES

Bluetooth SIG | www.bluetooth.com

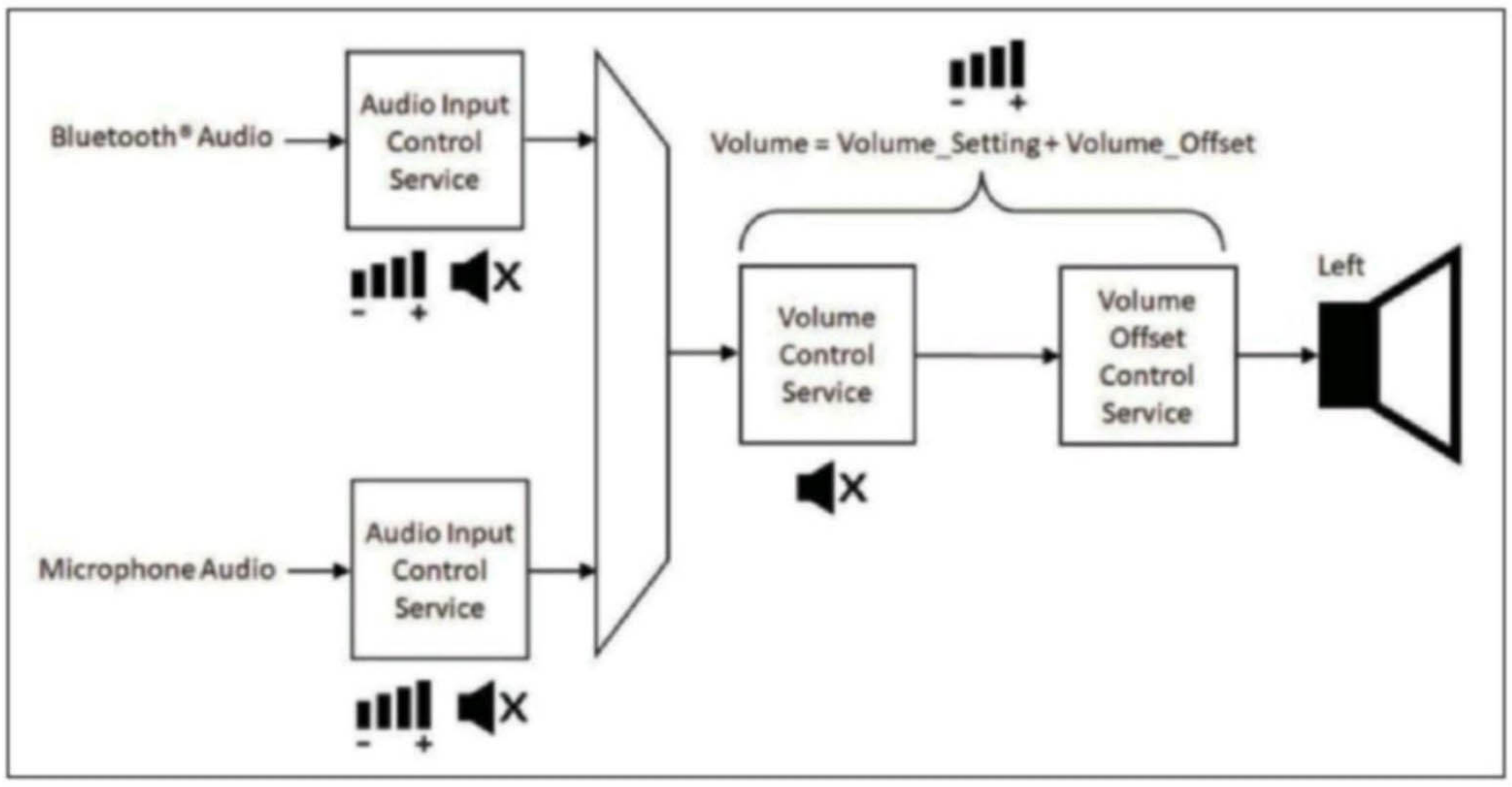

Bluetooth® Audio
Audio Input Control Service
- +
Microphone Audio
Audio Input Control Service
- +
- +
Volume = Volume_Setting + Volume_Offset
Volume Control Service
Volume Offset Control Service
Left

FIGURE 12
The elements of volume control.

confusion, the lock feature of CSIS and a Change_Counter within the volume services prevent clashes.

Figure 12 shows a typical example of how these specifications are used together in one of a pair of earbuds or hearing aids, illustrating the mains aspects that each controls. The other earbud or hearing aid would have an identical implementation, but its VOCS value would reflect the appropriate offset for the device in the right ear.

Finally, within GAF, the Microphone Input Control Profile and Service (MICP and MICS) allow individual control of microphones on Acceptors. It is increasingly common for devices to have multiple microphones to capture ambient sound as well as the wearer's voice. These specifications allow each input to be individually controlled in terms of gain, AGC and muting.

TOP LEVEL PROFILES

With all the complexity within the GAF, the Top Level Profiles don't introduce much new behavior. Instead, their main role is to mandate additional QoS requirements, typically in the form of higher sampling rates for the LC3 codec and shorter values of Presentation Delay. They target specific use cases. Devices that comply with them must support more of the optional features from within the GAF specifications, which may be LC3 parameters, or control features for volume, telephony and media.

Three of the Top Level specifications target specific use cases—the Hearing Access Profile (HAP); the Telephony and Media Profile (TMAP), which wraps up telephony and audio streaming into a single profile); and the Gaming Audio Profile (GMAP), which concentrates on low latency audio configurations. TMAP and GMAP have minimum service specifications included within their main profiles. The Hearing Access Service is a separate document, as it adds preset functionality which is specific to hearing aids. None of these three profiles tell a developer much about how Bluetooth LE Audio works—for that you need to understand how the underlying specifications work.

The final two Top Level profiles are the Public Broadcast Profile (PBP) and the Broadcast Audio URI (BAU) specification. Both of these are used to assist in the user experience of broadcast applications. The PBP defines how information about a broadcast stream should be used in the extended advertisements of a Broadcast Transmitter along with requirements on its use by scanning devices. It is a mandatory requirement for broadcast devices that comply with the Bluetooth SIG's Auracast guidelines, helping Broadcast Assistant devices to determine whether a broadcast stream is compatible with its Broadcast Receiver.

The BAU specification has little to do with the underlying specifications, but describes how the information describing a broadcast, which is defined in BASS, can be represented by a standardized URI. This allows a QR code to be used to identify a broadcast transmitter of a specific broadcast stream. Users can scan this with their phone, which will direct their earbuds to synchronize to a specific audio stream. It can also be used dynamically. For example, users wanting to share music streams from their phones can convert the stream to a broadcast, while displaying a QR code on their phone screens to allow their friends to share the same music.

IN CONCLUSION

That brings us to the end of this overview of the new Bluetooth LE Audio specifications and how they fit together. The broadcast capabilities, particularly those used for the new Auracast applications, transform the topology and usage models for audio. They're probably the most significant advance in audio since the introduction of stereo back in the 1950s. We now need developers to think outside the box of conventional audio applications to open up new ways to experience and share sound.

ABOUT THE AUTHOR

For the past 30 years ***Nick Hunn*** has been closely involved with short range wireless and communications, designing technology that helps to bring mobility to products, particularly in the areas of telematics, M2M, smart energy, IoT, LPWAN, wearables and mobile health. He coined the word "Hearables" and chaired the Bluetooth Hearing Aid working group, leading the development of LE Audio. Nick has published *An introduction to Bluetooth LE Audio* now in its second edition, and is also the author of *The Essentials of Short Range Wireless* a book attempting to explain the application of wireless technologies to product developers. He regularly writes about technology on his blog at www.nickhunn.com. He can be contacted at nick@wifore.com.

Create an AI-Driven Smart Home Security System

Part 1— Setup and First Steps of the Raspberry Pi AI Kit

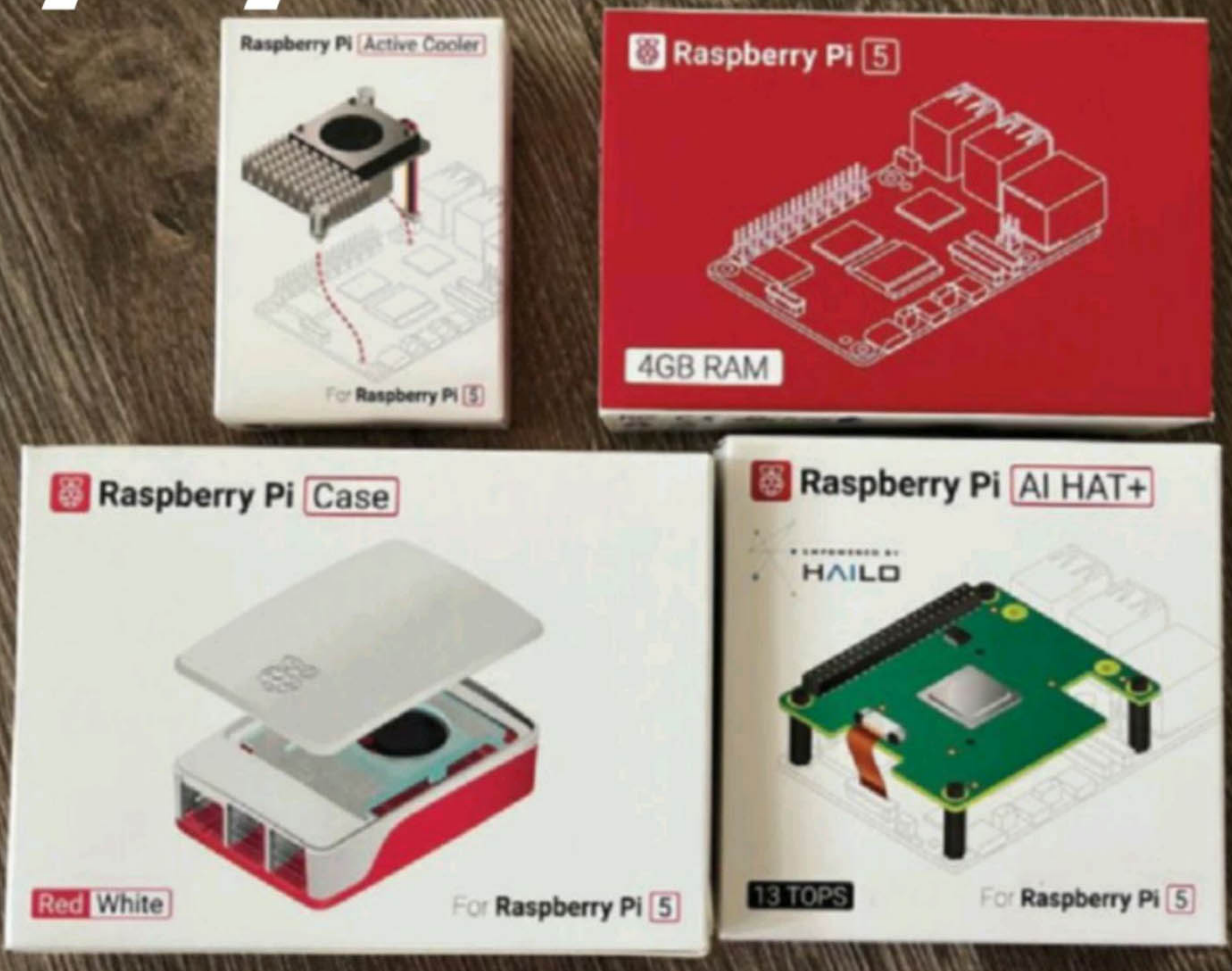


FIGURE 1
Major components required to build this project. Additionally, you will need a camera module (discussed later in the article).

By
Dhairya Parikh

Artificial intelligence has become a game-changer in various domains, and the Raspberry Pi community is no exception. In the first of this three-part series on creating a Smart Home Security System from scratch, Dhairya guides you through the essentials of setting up and testing a Raspberry Pi 5 with an AI Kit and camera module, and then running your first AI model.

With the introduction of tools based on large language models (LLMs) such as ChatGPT and Claude, artificial intelligence (AI) has taken center stage in recent years, pushing industries to adopt cutting-edge, AI-based solutions. This surge in AI innovation has trickled down to the consumer level, making it easier than ever to harness AI for everyday use.

Artificial intelligence has become a game-changer in various domains, and the Raspberry Pi community is no exception. To ride this wave of innovation, the Raspberry Pi AI Kit **(Figure 1)** is here to revolutionize the way we use Raspberry Pi for AI-powered applications. Developed by Hailo in collaboration with the Raspberry Pi team, this kit supercharges the Raspberry Pi, enabling it to handle sophisticated AI tasks with ease. This affordable and accessible hardware empowers tech enthusiasts and beginners alike to set up a reliable, AI-driven system right at home—leveraging object detection, real-time alerts, and seamless integration with smart devices. It's a practical step into the AI-powered future, demonstrating how AI can enhance safety and convenience in our daily lives.

Previously, running AI applications on the Raspberry Pi was a challenging task, because of its low-power CPU and GPU combination. The AI kit provides a dedicated GPU for AI-related tasks, making it an important component for anyone who uses Raspberry Pi for AI-based applications—specifically vision-based projects that use a camera and perform inferences based on what they see.

So, let's start! This month, I will focus on the basics—getting to know the Raspberry Pi AI Kit, setting it up with Raspberry Pi 5, and performing initial tests. You'll learn how to connect the Raspberry Pi AI Kit with the hardware provided, complete the necessary software installations, and integrate a camera (choose between the official Pi Camera or a USB camera). By the end, you'll run your first AI model to detect objects within the camera frame.

Future articles in this series will guide you through implementing advanced functionalities, such as detecting intruders, identifying faces, and recognizing packages—

creating a fully functional and powerful Smart Home Security System.

PROJECT HARDWARE

This section details all the hardware components we will be using for this project. Simply speaking, three main components are needed to build your very own home security system—Raspberry Pi, the Raspberry Pi AI Kit, and the Camera Module.

Raspberry Pi 5: The first and the most obvious piece of hardware you will need is the Raspberry Pi (Figure 1). More specifically, you need the latest Raspberry Pi 5 for this project. Your question now would be, why? I'll explain that before we move forward.

The Pi 5 is the only iteration that comes with a "PCIe interface," which is what the AI kit uses to connect the Hailo chip with the Pi. It features the most powerful processor available on a Pi, the Broadcom BCM2712. It also includes additional chips that were built by the great folks at Raspberry Pi. It uses a whopping 27W power supply, compared to the previous 15W used by the Pi 4.

So, Raspberry Pi 5 is the best and the only choice we have for this project. For those of you who are just getting started with Raspberry Pi, here are the technical specs for this versatile development board [1]:

- Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU, with cryptography extensions, 512KB per-core L2 caches and a 2MB shared L3 cache
- VideoCore VII GPU, supporting OpenGL ES 3.1, Vulkan 1.2
- Dual 4Kp60 HDMI display output with HDR support
- 4Kp60 HEVC decoder
- LPDDR4X-4267 SDRAM (2GB, 4GB, and 8GB)
- Dual-band 802.11ac Wi-Fi
- Bluetooth 5.0/Bluetooth Low Energy (BLE)
- microSD card slot, with support for high-speed SDR104 mode
- Two USB 3.0 ports, supporting simultaneous 5Gbps operation
- Gigabit Ethernet, with PoE+ support (requires separate PoE+ HAT)
- Two 4-lane MIPI camera/display transceivers
- PCIe 2.0 x1 interface for fast peripherals (requires separate M.2 HAT or any other adapter)
- 5V/5A DC power via USB-C, with Power Delivery support

Raspberry Pi/Hailo AI Kit: This is the hardware that will do all the work in our project (Figure 1). Although it is currently possible to run some simple AI models on the Pi without this kit, the performance output would be pretty limited You would get only 5fps to 10fps when you run a small YOLO (You Only Look Once) model on the onboard CPU and GPU. (Don't worry if you are seeing the term "YOLO" for the first time; this model will be covered extensively throughout this series.) Sophisticated and advanced hardware is required to run these advanced models, which need a lot of computation under the hood.

That is exactly what the Raspberry Pi AI Kit provides—a cost-effective way to run some complex AI models on the card-sized Raspberry Pi. Pretty fascinating, right? Let's learn more about the kit to get a better idea of what we get:

Three versions of the kit are available on the market right now. In the first one, you get a Raspberry Pi M.2 HAT+ with a Hailo AI Acceleration Module (fitted with the Hailo 8L module, capable of running 13TOPS (tera-operations per second). Apart from this, there are two dedicated Hailo kits, which come with an embedded Hailo acceleration chip (not a module), capable of going 13 and 26TOPS, respectively.

Each kit comes with the following contents:

- Hailo AI module containing a Neural Processing Unit (NPU)
- Raspberry Pi M.2 HAT+, to connect the AI module to your Raspberry Pi 5
- Thermal pad pre-fitted between the module and the M.2 HAT+
- Mounting hardware kit
- 16mm stacking GPIO header

Each version of the kit communicates with the Raspberry Pi through the superfast PCIe interface for best performance. **Figure 2**

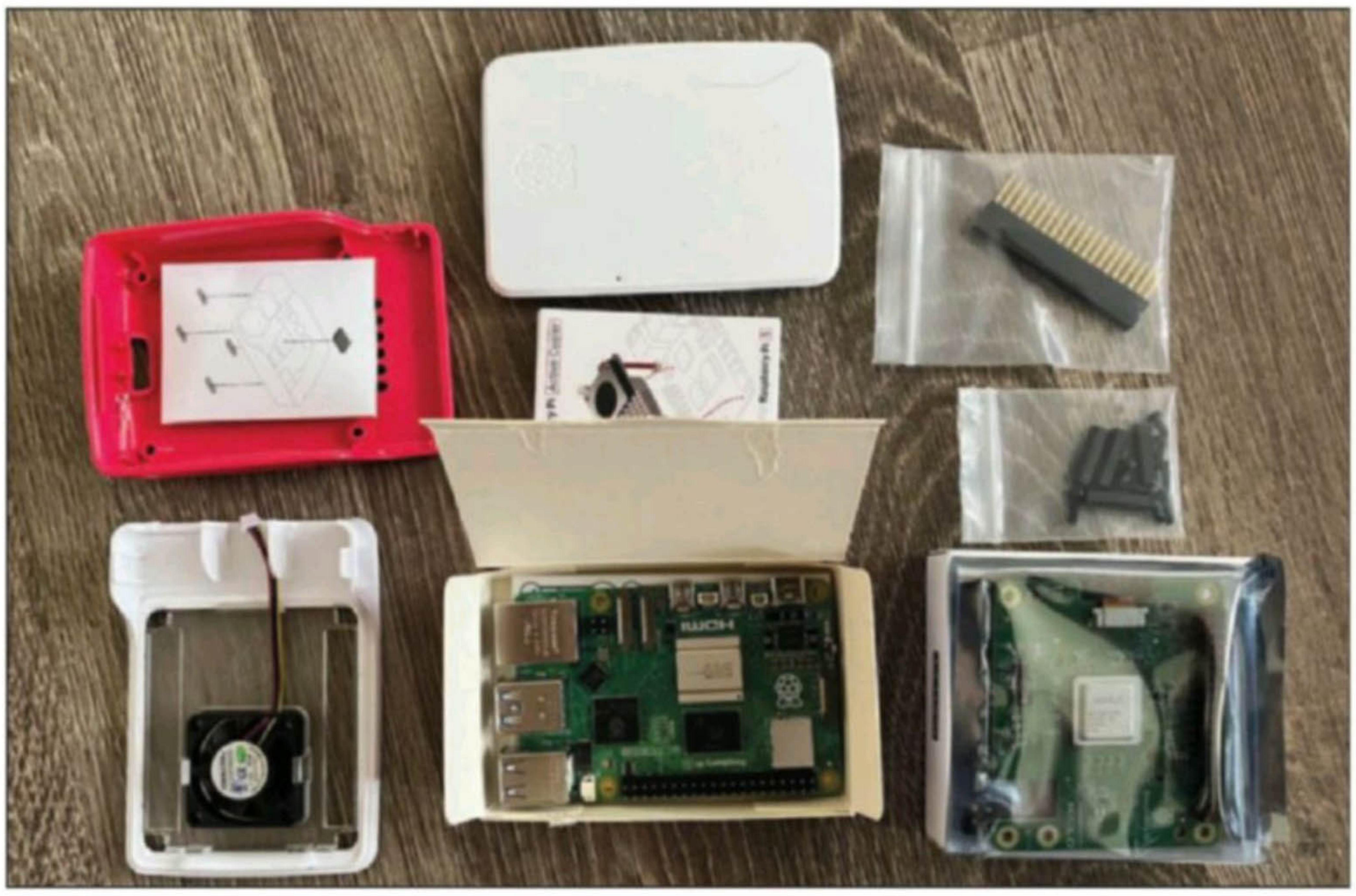


FIGURE 2
A better look at the components that will be used to build the project: a Raspberry Pi 5, the AI Kit, and additional attachments such as the official Pi case and cooler.

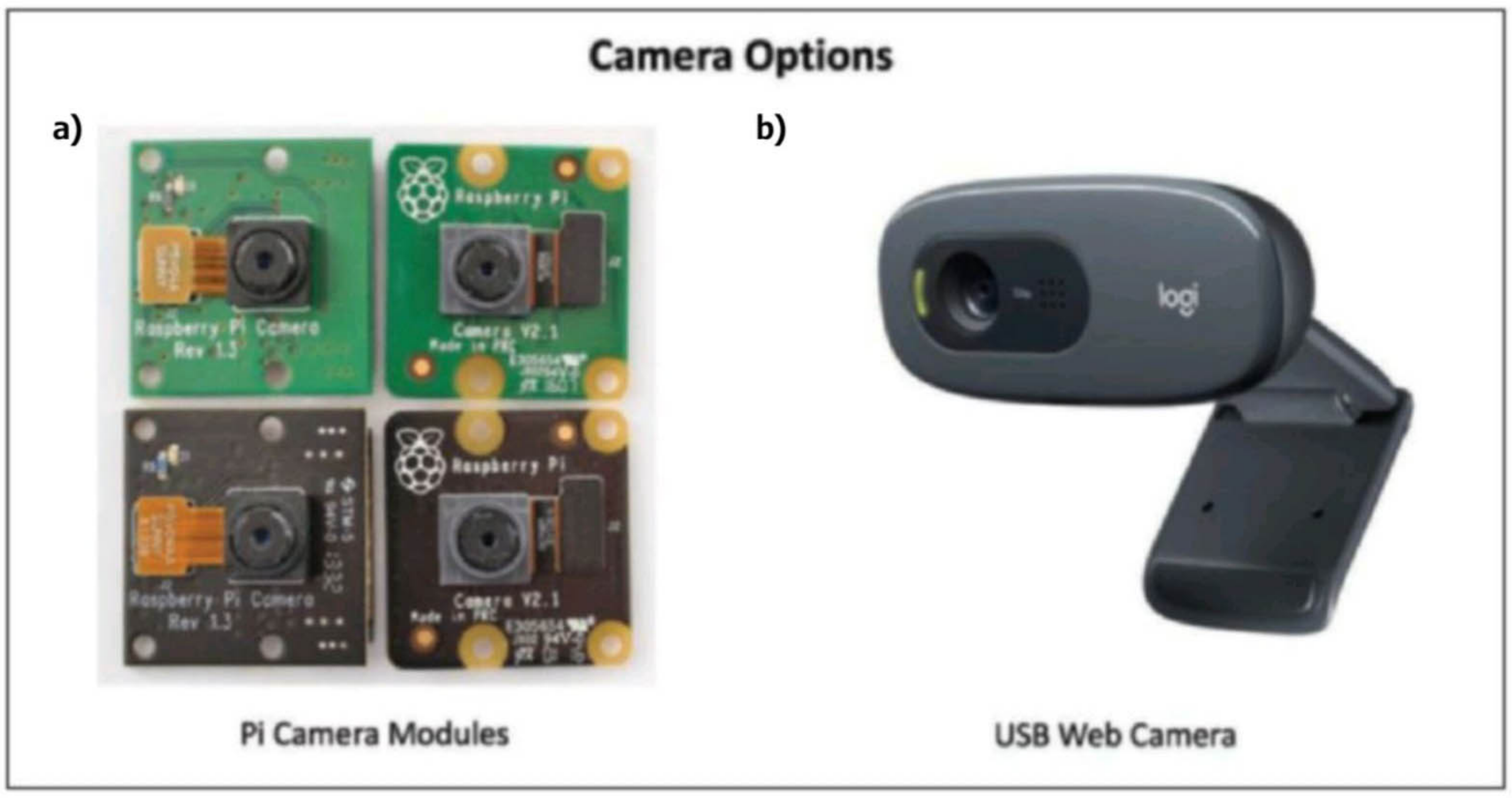


FIGURE 3
Camera options for this project. You can either opt for an official Pi camera module, a trusted manufacturer's clone of it, or a generic USB camera module [3].

gives you a better look into the components of the kit.

Camera Module: Since our home security system is vision based, we will need a camera module. I strongly suggest that you use one of the official Pi camera modules (**Figure 3a**) available on the market. They are pretty affordable and have really good support when it comes to technical difficulties. Alternately, if you are tight on budget and you already have an old USB webcam (**Figure 3b**), you can use that for the project as well.

It is important to note that if the USB camera is not from a known manufacturer, it may not be detected by the Pi. I will go deeper into the camera setup process later in this article. As before, let's compare the two options I presented to you, so that you can make an informed choice:

1) Raspberry Pi Camera Module (the latest version)
    - 12.3-megapixel Sony IMX477 sensor
    - Support for 4K30 video recording
    - Wide dynamic range
    - Excellent low-light performance
    - Direct connection via CSI port

2) USB Camera
    - Any standard USB webcam (1080p recommended)
    - Plug-and-play compatibility
    - Flexible positioning options
    - Often more cost-effective than the first option.

Once you have chosen the type of camera, you are ready to start the setup process. I will go through this step by step, starting with readying the Raspberry Pi 5 for the AI kit installation.

HARDWARE SETUP

In this section, I will walk you through the process of setting up the Raspberry Pi hardware and the AI Kit. I will go through the entire process in three simple steps. Let's start by readying our Raspberry Pi for the AI kit.

Step 1: Setting up the Raspberry Pi 5: Make this a rule. Whenever you are starting a new project with your Pi, always make sure that it is running the latest version of the software. This will save you from a lot of potential issues down the line. You can do this by running this single command:

```
sudo apt update && sudo apt full-upgrade
```

You might have to restart your Pi for the updates to take effect. Once this is done, rerunning the above command should show that the software is up to date, which should confirm that your Pi is running on the latest software:

0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded. (command output preview)

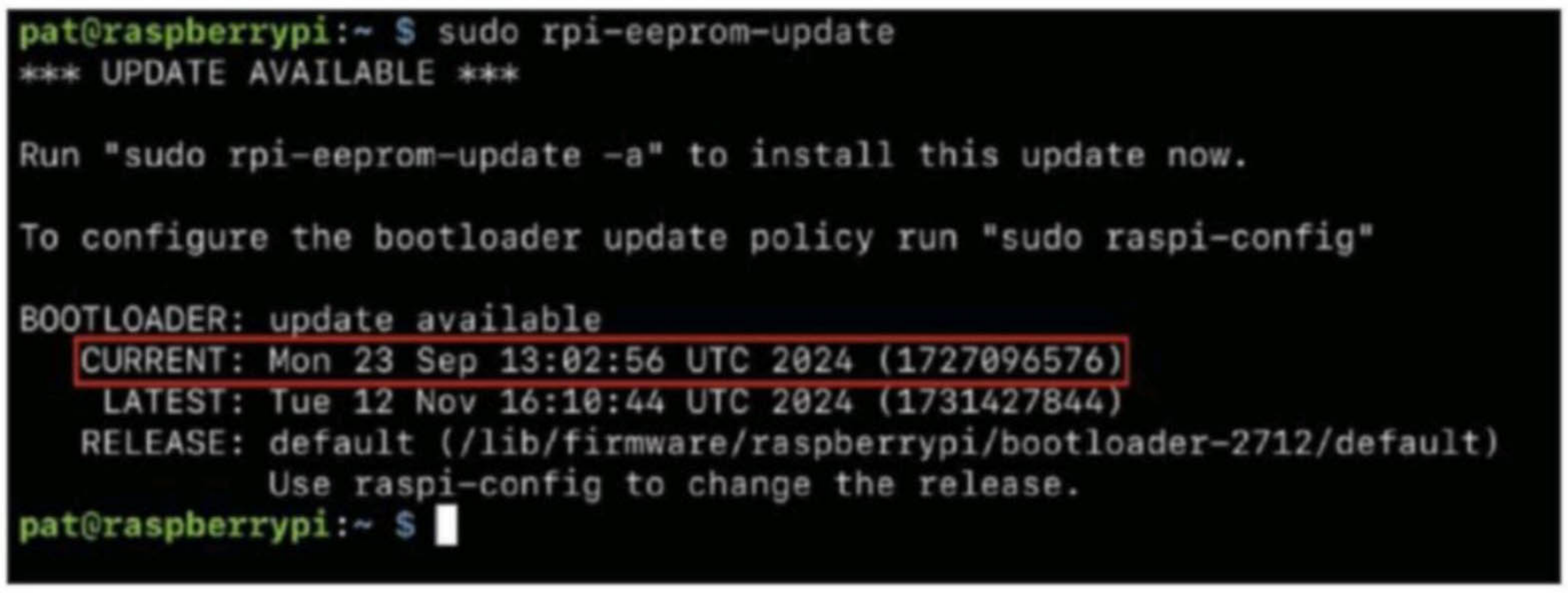
Another crucial aspect to check is the Raspberry Pi Firmware version. The AI Kit requires a Firmware version later than December 6, 2023. You can check the version your Pi is running using the following command (see **Figure 4** for output).

```
sudo rpi-eeprom-update
```

If you see an earlier date, you will have to update it to a newer version. (I suggest going with the latest version.) Please follow the steps given in the official Raspberry Pi guide for bootloader updates [2].

Once this is complete, you are ready to install the AI Kit hardware onto the Raspberry Pi!

Step 2: Installing the Raspberry Pi AI Kit: The AI kit comes with the AI module, which can be separate if it's the older version, or soldered to the HAT if it is the HAT+, which is the newer iteration of the product. The kit also includes a ribbon cable, a GPIO header stack, and the required mounting hardware.



```
pat@raspberrypi:~ $ sudo rpi-eeprom-update
*** UPDATE AVAILABLE ***

Run "sudo rpi-eeprom-update -a" to install this update now.

To configure the bootloader update policy run "sudo raspi-config"

BOOTLOADER: update available
   CURRENT: Mon 23 Sep 13:02:56 UTC 2024 (1727096576)
    LATEST: Tue 12 Nov 16:10:44 UTC 2024 (1731427844)
   RELEASE: default (/lib/firmware/raspberrypi/bootloader-2712/default)
            Use raspi-config to change the release.
pat@raspberrypi:~ $
```

FIGURE 4
Output for the EEPROM command execution, showing the current version of the firmware, along with the latest available version. For our AI kit, we need a firmware with a date later than December 6, 2023.

The folks at Raspberry Pi have created a really nice and visual step-by-step tutorial on how to install the AI Kit onto your Raspberry Pi [3]. Once you have installed the AI Kit, your Pi will look as shown in the last step of the tutorial.

One thing to note is that their tutorial also includes a step of installing the Active Cooler onto the Pi. It is highly recommended when performing compute-intensive tasks on the Pi, but it's not mandatory.

Step 3: Setting up the Camera: Now that you have the AI Kit installed and ready, you can proceed with the setup process for this kit. As previously mentioned, you will need a supported camera for this project. So, whichever camera you choose, just connect it to your Raspberry Pi. There are two ports for the official camera modules; keep in mind that you will need an additional cable if you are using the older version of the camera module, because the new port is different. If you are using a USB Web Camera, you can just connect it to one of the USB ports.

One important thing to note is to always disconnect the Pi from power before inserting or removing the camera module, since doing so with the power connected can destroy your camera module. I say this from experience.

Once you have connected the camera module, you should have a module that looks something like **Figure 5**. Now, connect the Pi to power, and let's start the software setup.

SOFTWARE SETUP

You need a few packages to get your Hailo module integrated with the Pi. Moreover, you will also run your very first module on the Raspberry Pi!

1) Before you start, an optional step is recommended by the folks at Raspberry Pi to get the most out of your AI Kit—enabling the PCIe Gen 3.0 interface (disabled by default). You can easily do this from the Raspberry Pi configurations as detailed below.

    - From the main menu, first select "Advanced Options."
    - Then select "PCIe Speed."
    - When prompted, choose "Yes" to enable PCIe Gen 3 mode.
    - Finally, select "Finish" to exit, and press "Enter."
    - You will have to reboot your Raspberry Pi with `sudo reboot` for your changes to take effect.

    Be aware that the Raspberry Pi 5 hardware is not officially certified for the speeds that the Gen 3.0 interface offers, so if you are facing problems connecting with your kit, it is best to disable it. (I personally haven't faced any problems yet.)

2) The first things you need are the dependent software packages required to use the AI Kit's NPU (the fancy Hailo chip). That includes drivers, firmware, and packages needed for operations. This can be done by executing the following command in a terminal window:

```
sudo apt install hailo-all
```

3) Once this is complete, you will need to reboot the Pi for the new settings to take effect.

4) To verify that the installations were successful, execute the following command on the terminal:

```
hailortcli fw-control identify
```

    If you did everything correctly, you should see an output like this:

```
Executing on device: 0000:01:00.0
Identifying board
Control Protocol Version: 2
Firmware Version: 4.17.0
(release,app,extended context switch
buffer)
Logger Version: 0
Board Name: Hailo-8
Device Architecture: HAILO8L
Serial Number: HLDDLBB234500054
Part Number: HM21LB1C2LAE
Product Name: HAILO-8L AI ACC M.2 B+M
KEY MODULE EXT TMP
```

The highlighted options may show a value of N/A if you are using the newer AI HAT+ (the model where the Hailo chip is embedded onto the board, not an extension).

This completes the setup process for this kit. You are now ready to configure and run

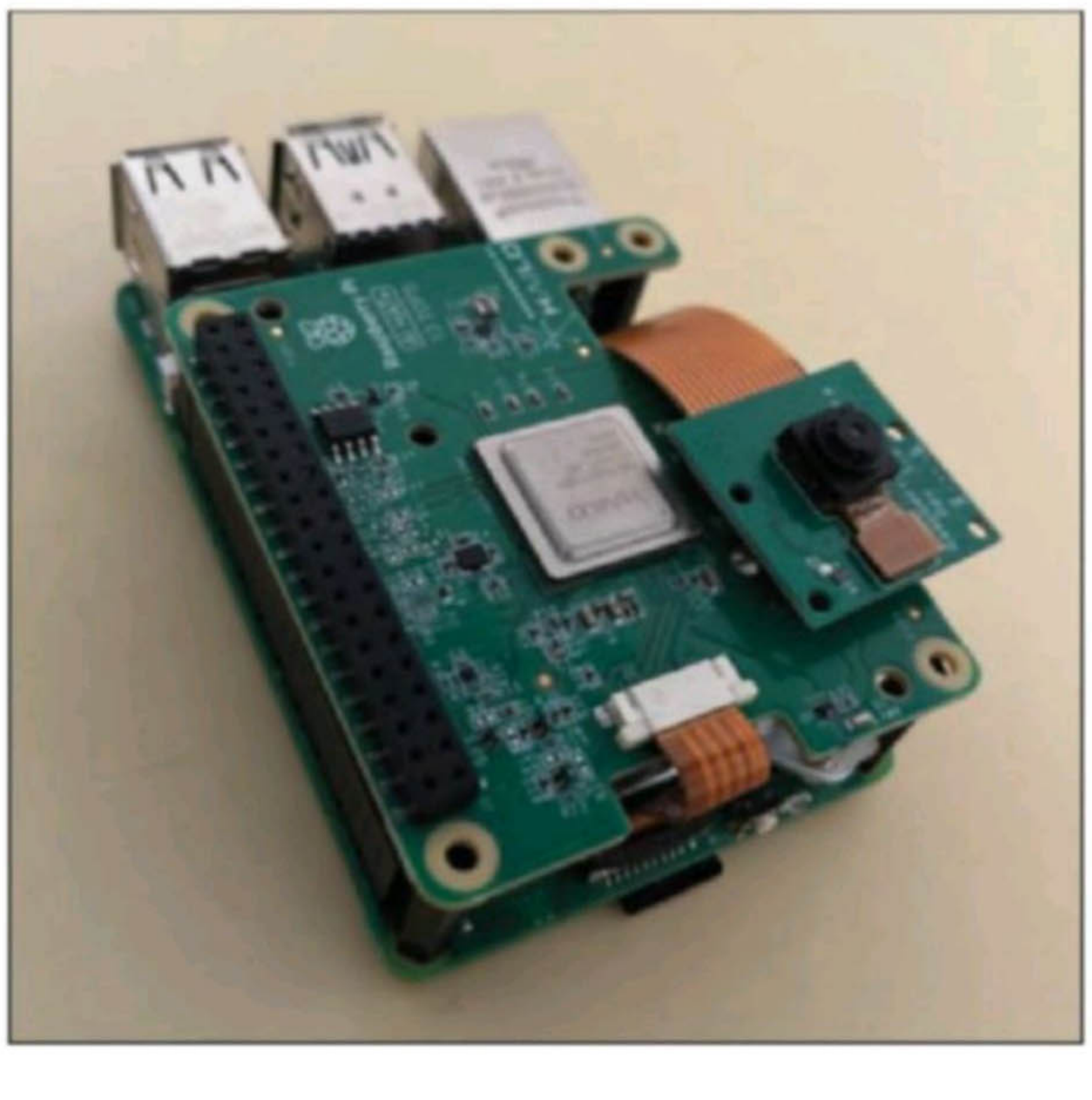
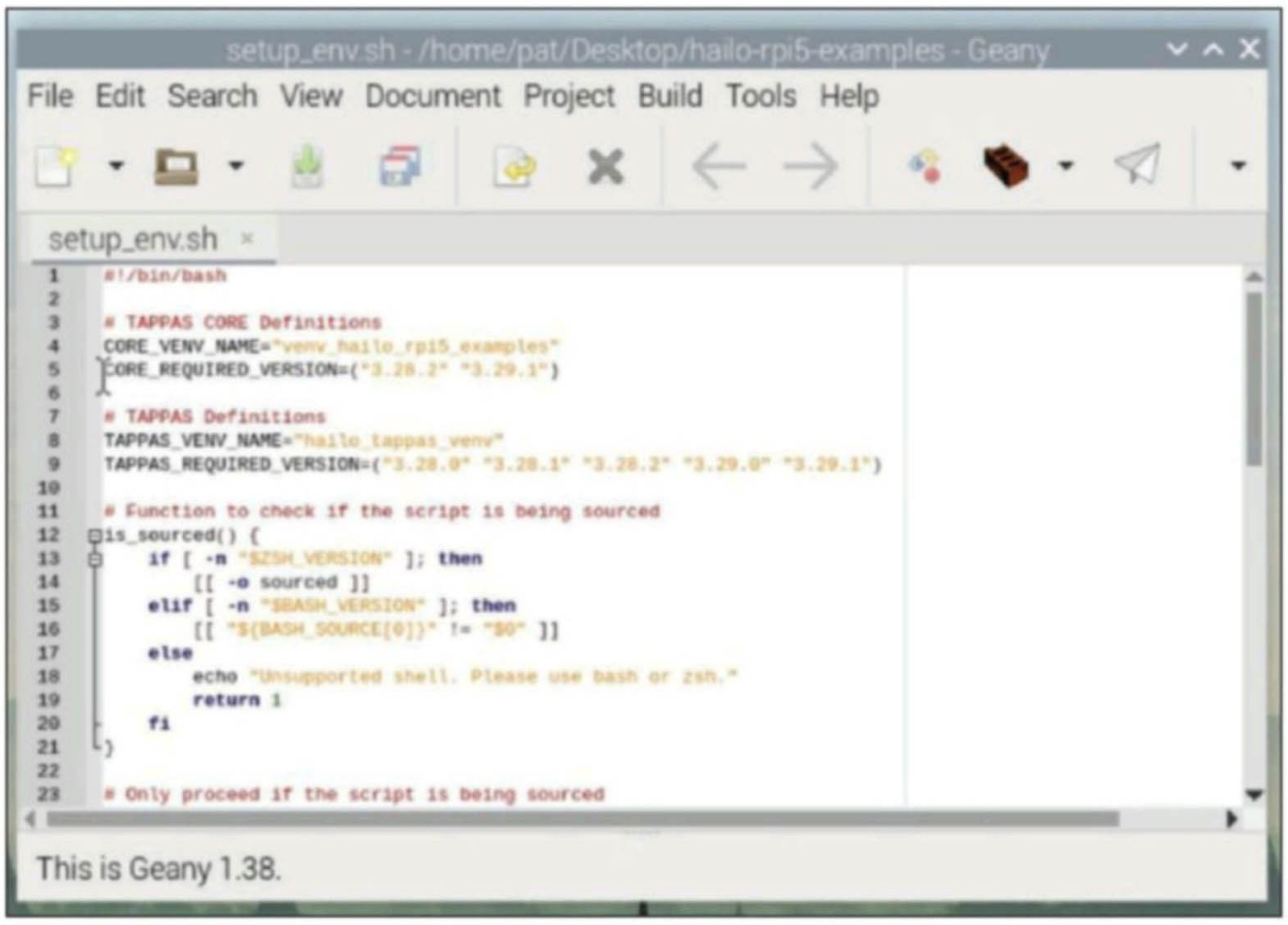


FIGURE 5
A look at the assembled hardware (without the Pi case). This includes the active cooler, the AI kit, and the Pi camera, with everything mounted on the Raspberry Pi 5.


setup_env.sh - /home/pat/Desktop/hailo-rpi5-examples - Geany
File Edit Search View Document Project Build Tools Help
setup_env.sh

```
#!/bin/bash

# TAPPAS CORE Definitions
CORE_VENV_NAME="venv_hailo_rpi5_examples"
CORE_REQUIRED_VERSION=("3.28.2" "3.29.1")

# TAPPAS Definitions
TAPPAS_VENV_NAME="hailo_tappas_venv"
TAPPAS_REQUIRED_VERSION=("3.28.0" "3.28.1" "3.28.2" "3.29.0" "3.29.1")

# Function to check if the script is being sourced
is_sourced() {
    if [ -n "$ZSH_VERSION" ]; then
        [[ -o sourced ]]
    elif [ -n "$BASH_VERSION" ]; then
        [[ "${BASH_SOURCE[0]}" != "$0" ]]
    else
        echo "Unsupported shell. Please use bash or zsh."
        return 1
    fi
}

# Only proceed if the script is being sourced
```

This is Geany 1.38.

FIGURE 6
A snapshot of the setup_env.sh file, giving us a better look at the commands that the file runs to set various environment variables and create a Python virtual environment.

AI models on your Raspberry Pi! Before you do that, let's ensure that your camera is connected and working:

TESTING THE CAMERA

Raspberry Pi Camera: For official camera modules, you can verify that they are running by executing the following command:

```
rpicam-hello -t 5s
```

This command will start the camera and show a preview for it on the Pi's screen for 5 seconds.

USB Camera: For these cameras, you can execute the following command and check for an entry listed "/dev/video(n)," where "n" is the ID of the camera with its value starting with "0":

```
ls /dev/
```

Once this is done, you are ready to start with the first setup for your project, and run your first AI Model on the Raspberry Pi using the AI kit.

RUNNING YOUR FIRST AI MODEL

There are several ways to run AI models on the Pi using the AI kit, all thanks to the great development tools provided by Hailo's technical team. For this tutorial series, we will be using the "hailo-rpi5-examples" GitHub repository [4], which is a collection of ready-to-use, real-world examples for the AI Kit. We will use this as a reference and build upon it.

It will include some additional steps, but you will see that it is worthwhile, because it saves a lot of time and also offers numerous customization options. Let's get started! Just follow the steps below, and you'll be running an object detection AI model on your Pi within 15 minutes.

Step 1: Cloning the Repository: First, you will need the code presented in the GitHub repository [4], which you have to clone on your Raspberry Pi. If this is new for you, think of cloning as downloading the files from the Internet onto your system. You can do so by executing the following command from a terminal window:

```
cd Desktop && git clone
https://github.com/hailo-ai/hailo-
rpi5-examples.git
```

This will create a new folder on your Pi's desktop and you'll see all the code included in the repository there.

Step 2: *Setting up the Python Environment:* This repository uses the Python programming language. Python comes pre-installed as part of the Raspberry Pi OS package. So, you now need to install and set up the Python installations, so you can run the examples provided in this repository. Fortunately, the authors of this repository provided a file that is a collection of commands that does this. It's named setup_env.sh in the repository (see **Figure 6**).

To run this file, execute this command in the same terminal window:

```
cd hailo-rpi5-examples && source
setup_env.sh
```

This script will set all the required environment variables and create a new Python virtual environment. Now, you need to install all the necessary packages in this environment. For this, execute the following command:

```
pip install -r requirements.txt
```

This will take a few minutes to complete. You may be asked for a "Yes" or "No" to proceed with the installation, so press "Y "and then "Enter" to start the installation process.

Once this is done, execute the following command so that the rapidjson package is installed on your Pi. (It currently comes

Additional materials from the author are available at:
www.circuitcellar.com/article-materials

References [1] to [4] as marked in the article can be found there.

RESOURCES

Raspberry Pi Foundation | www.raspberrypi.org

Hailo | www.hailo.ai

preinstalled with the Raspberry Pi OS package, but that may change in the future).

```
sudo apt install -y rapidjson-dev
```

Next, you will need some pretrained models that will be run as part of the examples. They are categorized as resources, and to install the ones needed for this tutorial, just execute this command:

```
./download_resources.sh
```

This will download all the necessary AI models for this tutorial. It may take time, depending on your Internet speed.

Once the download is complete, some additional files required to run the demos are needed. For this, just run the following command:

```
./compile_postprocess.sh
```

This completes all the required steps to run the model. Now, let's see the AI kit in action, as the very first object detection model is run on the Raspberry Pi.

Note that you will have to run this command every time you want to run an example from this repository, because the code activates a Python virtual environment where everything is installed. However, the initial setup code will only be run once.

Step 3: Running the First Model: We are finally ready to run our first AI model on the Pi. Thanks to the examples provided with the repository, we can do this in a few simple steps:

1) If you are using a USB Camera, use the following command to run a Python script. It will help you identify the all the available USB cameras, and give you a reference number for your camera—generally "video0" in most cases).

```
python basic_pipelines/get_usb_
camera.py
```

2) Depending on the type of camera you are using, you will have to execute one of the two commands listed below, depending on which camera you have.

```
python basic_pipelines/detection.
py -input rpi (for Pi Camera
Modules)
python basic_pipelines/detection.
py -input /dev/video<x> (For USB
Cameras)
```

A few seconds after running the command, you should see the camera output on your Pi's home screen, with the object detection model running on it. You will be able to see the detected objects logged in the terminal, along with the frame rate, as shown in **Figure 7**. I got a steady frame rate of 30fps, which is impressive, given that I was using a first-generation Pi camera, and typically, running the same model on Pi's CPU/GPU results in a 5fps to 10fps performance.

CONCLUSION AND NEXT STEPS

We now have a fully setup Raspberry Pi with our AI kit installed and have already tested our first AI model on it. Thus far we have discussed three main points in this tutorial. We have provided an overview of the hardware (the Raspberry Pi AI Kit, Raspberry Pi 5, and the camera modules). We then set up the hardware and the software for the AI Kit. And finally we ran the first AI model on the Pi + AI Kit combo.

In the Part 2 of this series, we will start working on the actual use case—building your very own home security system. More specifically, you will learn how to add the primary functionality of Intruder Detection, and test its functionality.

Thank you for reading this article. See you in the next one!

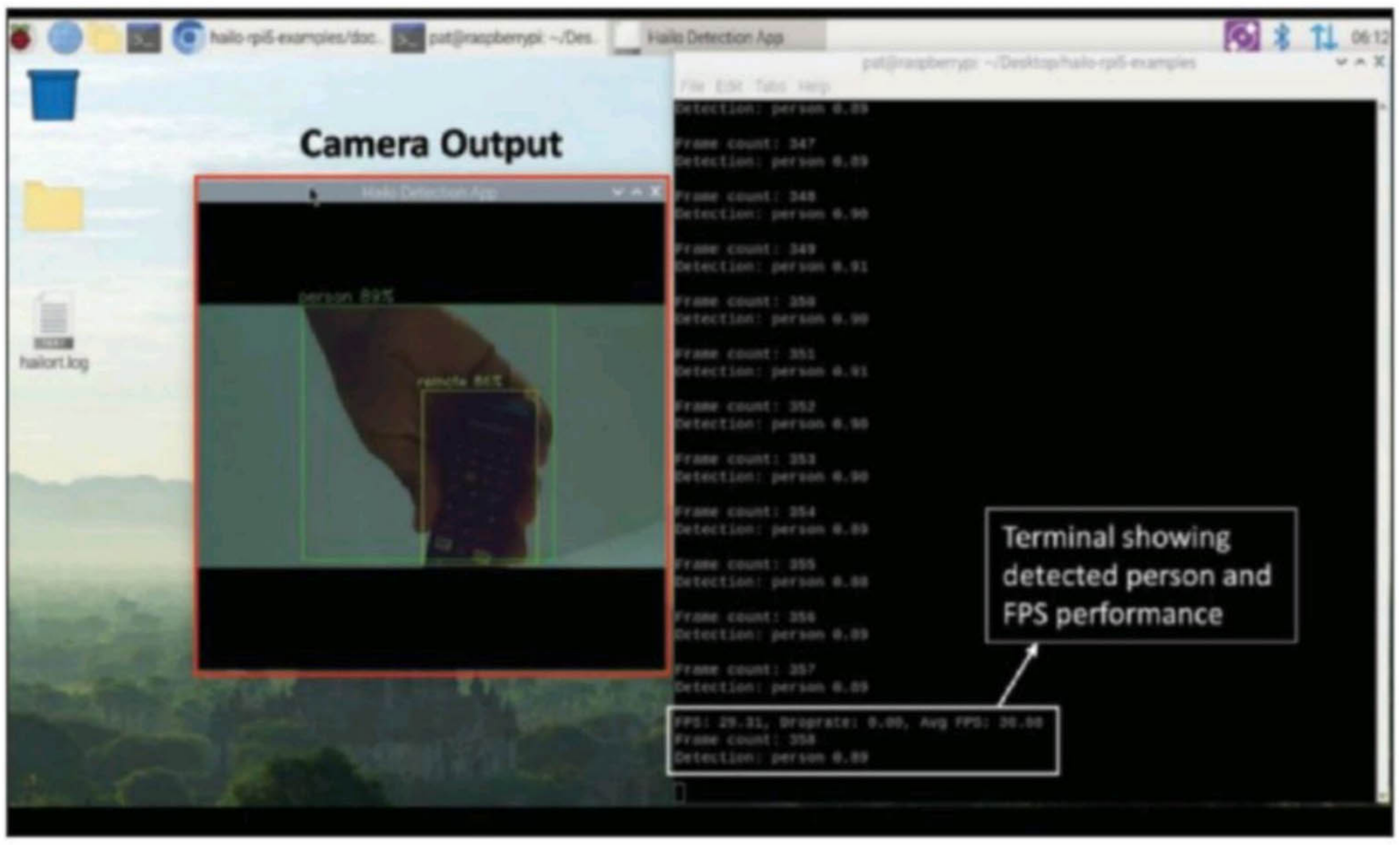


FIGURE 7
Object detection model successfully running on the Raspberry Pi with the AI kit. Along with the camera output, you can also see the terminal output with the frame rate and the detection of the object labeled "person"—which in this case is my hand.

ABOUT THE AUTHOR

Dhairya Parikh is a seasoned Technical Writer with more than five years of expertise in IoT, Machine Learning, and Hardware. He has authored a book on Raspberry Pi, published numerous technical articles, and currently contributes to RaspberryTips, a leading Raspberry Pi blog. With a talent for simplifying complex topics, Dhairya creates engaging step-by-step guides for developers and tech enthusiasts. He has completed his master's in computer engineering from the University of Waterloo, specializing in AI and Machine Learning.

Industrial Automation

Revolutionizing Production with Innovation

By
Caleb Smith

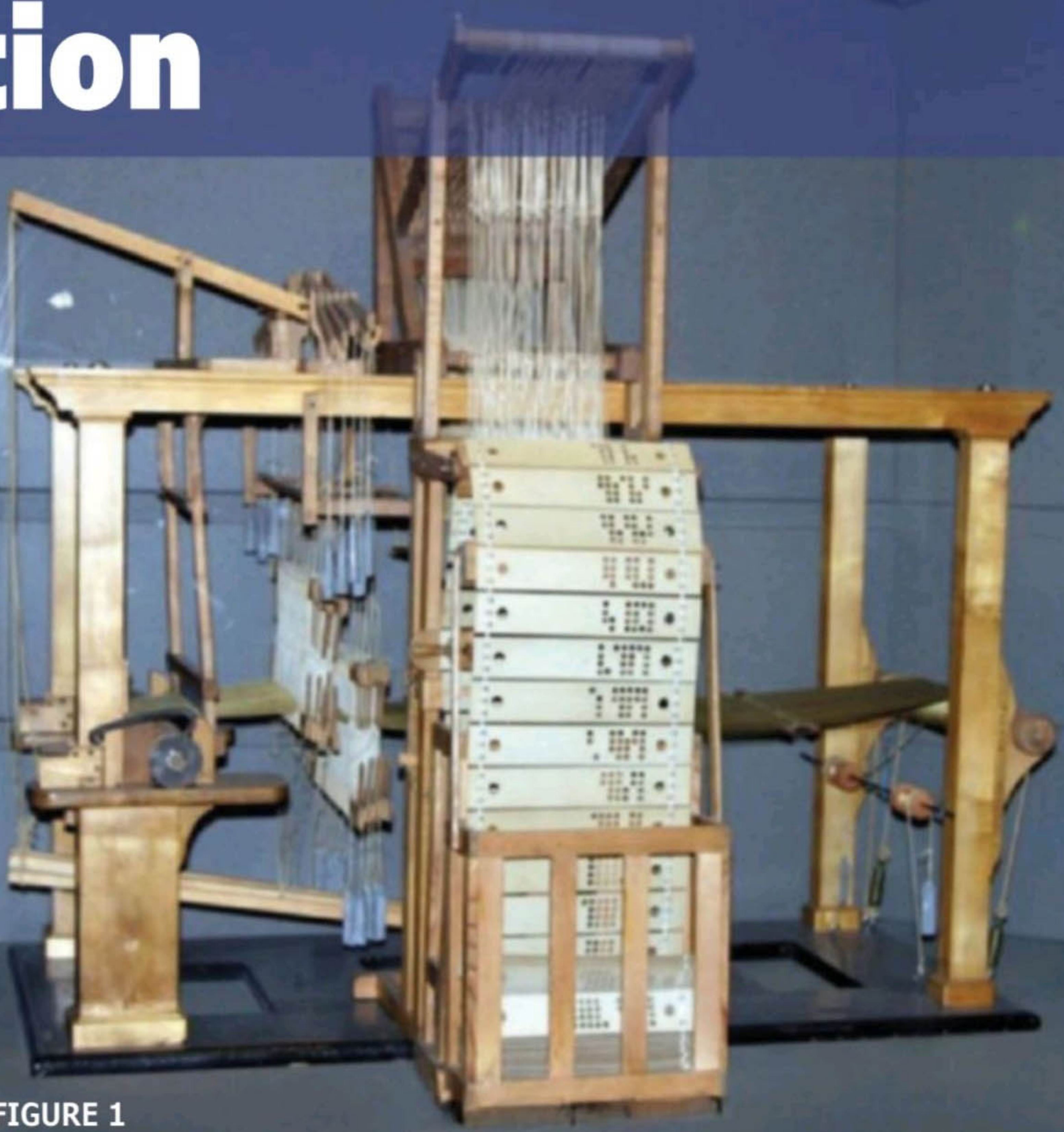


FIGURE 1
A modern recreation of Joseph Marie Jacquard's system of punch cards being used to program an automatic loom. The first automated industrial process.

Dive into the evolution of automation, from the groundbreaking inventions of the 18th century to today's cutting-edge technologies such as IoT and AI. Discover how these advancements are reshaping industries, enhancing efficiency, and posing new ethical challenges in a world increasingly reliant on smart systems.

Automation is the process by which a task is accomplished without the need for manual labor on the part of humans. From cooking toast in a toaster to the assembly of microchips and automobiles on factory assembly lines, much of modern life has become automated. This trend, however, is nothing new, and has been in the making for hundreds of years. In 1745, a French inventor named Jacques de Vaucanson created the first automated loom. Fifty-five years later in 1800, Joseph Marie Jacquard, a weaver, invented a system of punch cards that allowed patterns to be saved and programmed into looms (**Figure 1**), creating what was in essence, the first programmable computer. Less than 250 years later, we can store billions of times the amount of data as a stack of these punch cards onto silicon chips the size of finger nails and perform trillions of calculations per second, allowing for the creation of everything from automobiles and computers to medical devices and weapons.

INTERNET OF THINGS

The Internet of Things (IoT) is a network of interconnected "smart devices" capable of collecting and exchanging data for enhanced collaboration and analysis. While the term "Internet" might not always be directly involved, a communication network is essential for information sharing. This network can range from simple Bluetooth connections between a smartwatch and smartphone to more complex industrial networks connecting machines across a factory floor. The convenience of IoT extends beyond user-centric applications, revolutionizing industries such as manufacturing.

In the realm of manufacturing, IoT devices empower communication between disparate machines, facilitating rapid production cycles and intricate network control. These interconnected systems enable real-time data collection and analysis, leading to optimized workflows, predictive maintenance, and enhanced overall efficiency.

To cater to the unique demands of IoT applications, specialized chips have emerged, designed for low latency and real-time performance. Take, for example, the Texas Instruments C2000 series, specifically engineered for automotive and industrial applications. These chips boast increased power density and switching frequency, enabling them to handle demanding tasks with minimal delay.

Another notable example is the SAM L11 family of microcontrollers, renowned for their exceptional low-power consumption. These 32MHz chips prioritize security peripherals without sacrificing battery life, making them ideal for applications where power efficiency is paramount. The SAM L11 family features up to 64kb of flash memory and 16kb of SRAM, available in both 24- and 32-pin configurations. Furthermore, the SAM L11-KPH family incorporates a factory-provisioned root of trust key, guaranteeing its identity in IoT applications and bolstering security measures (**Figure 2**).

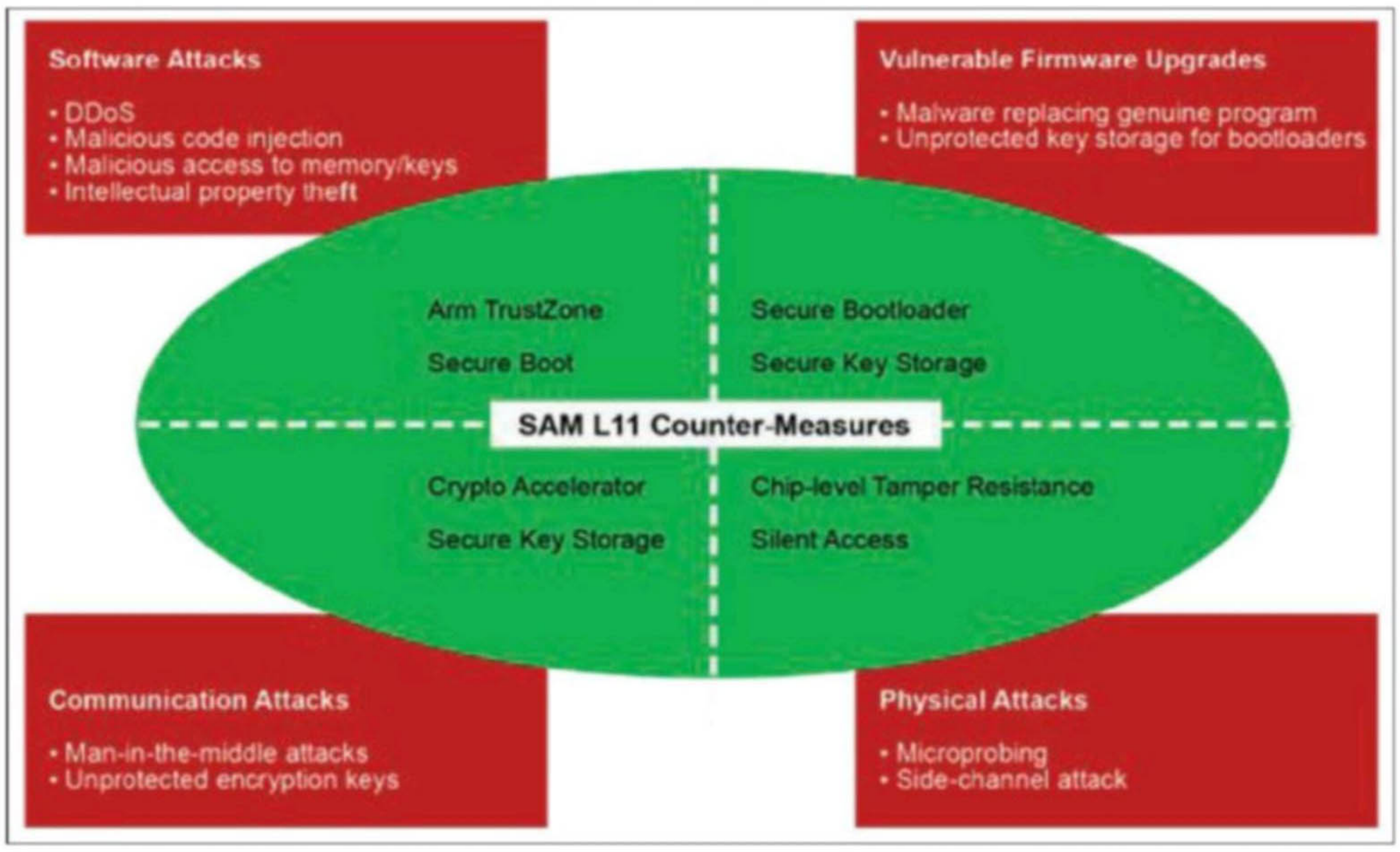

Software Attacks
• DDoS
• Malicious code injection
• Malicious access to memory/keys
• Intellectual property theft
Vulnerable Firmware Upgrades
• Malware replacing genuine program
• Unprotected key storage for bootloaders
Arm TrustZone
Secure Boot
Secure Bootloader
Secure Key Storage
SAM L11 Counter-Measures
Crypto Accelerator
Secure Key Storage
Chip-level Tamper Resistance
Silent Access
Communication Attacks
• Man-in-the-middle attacks
• Unprotected encryption keys
Physical Attacks
• Microprobing
• Side-channel attack

FIGURE 2
SAM L11 built-in security measures against physical and non-physical attacks.

While not all devices are inherently IoT-compatible, solutions such as Intel Smart Connect IoT Gateway Software bridge the gap, enabling any Intel x86 device to seamlessly connect with various cameras, sensors, and equipment. This software facilitates data collection, storage, and upload to cloud-based servers for comprehensive analysis. The "plug and play" setup requires minimal configuration, making it accessible for both small businesses and large enterprises to create custom IoT devices. This flexibility allows for tailored solutions that meet individual needs, ensuring local and secure management of IoT deployments.

The transformative potential of IoT extends beyond individual devices and software solutions. The interconnected nature of IoT systems fosters a dynamic ecosystem where data flows freely, enabling intelligent decision-making and continuous improvement. This interconnectedness fuels the development of advanced analytics, machine learning algorithms, and predictive models, further enhancing the capabilities of IoT applications.

As we move forward, the integration of artificial intelligence (AI) with IoT is poised to reshape industries, driving automation to unprecedented levels. AI-powered IoT systems can analyze vast amounts of data, identify patterns, and make autonomous decisions, leading to more efficient operations, reduced costs, and enhanced safety.

The future of IoT is brimming with possibilities, promising to revolutionize industries and redefine our interactions with technology. As we continue to harness the power of interconnected devices, the world is poised to become more intelligent, automated, and interconnected than ever.

ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) is arguably one of the most significant advancements in computer science this decade, capturing widespread attention and finding applications across a diverse range of fields. From analyzing large datasets to enhancing user interfaces, and from generating custom prompts to creating images, AI has become an integral part of modern technology. Over the past three years, its adoption has surged in nearly every technology-related sector, and there are no indications that this growth will slow down anytime soon.

In the realm of automation, AI plays a crucial role by analyzing existing processes to identify inefficiencies and improve workflow and performance. This capability allows businesses to streamline operations, reduce costs, and enhance productivity. Moreover, AI is increasingly being utilized to automate tasks that were traditionally considered exclusive to human capabilities, such as writing content or engaging in customer communications. This shift not only frees up human resources for more complex tasks but also enhances the speed and efficiency of service delivery.

Despite the numerous resources available for leveraging AI in both business and personal contexts, many individuals and organizations face challenges when it comes to locally hosting machine learning

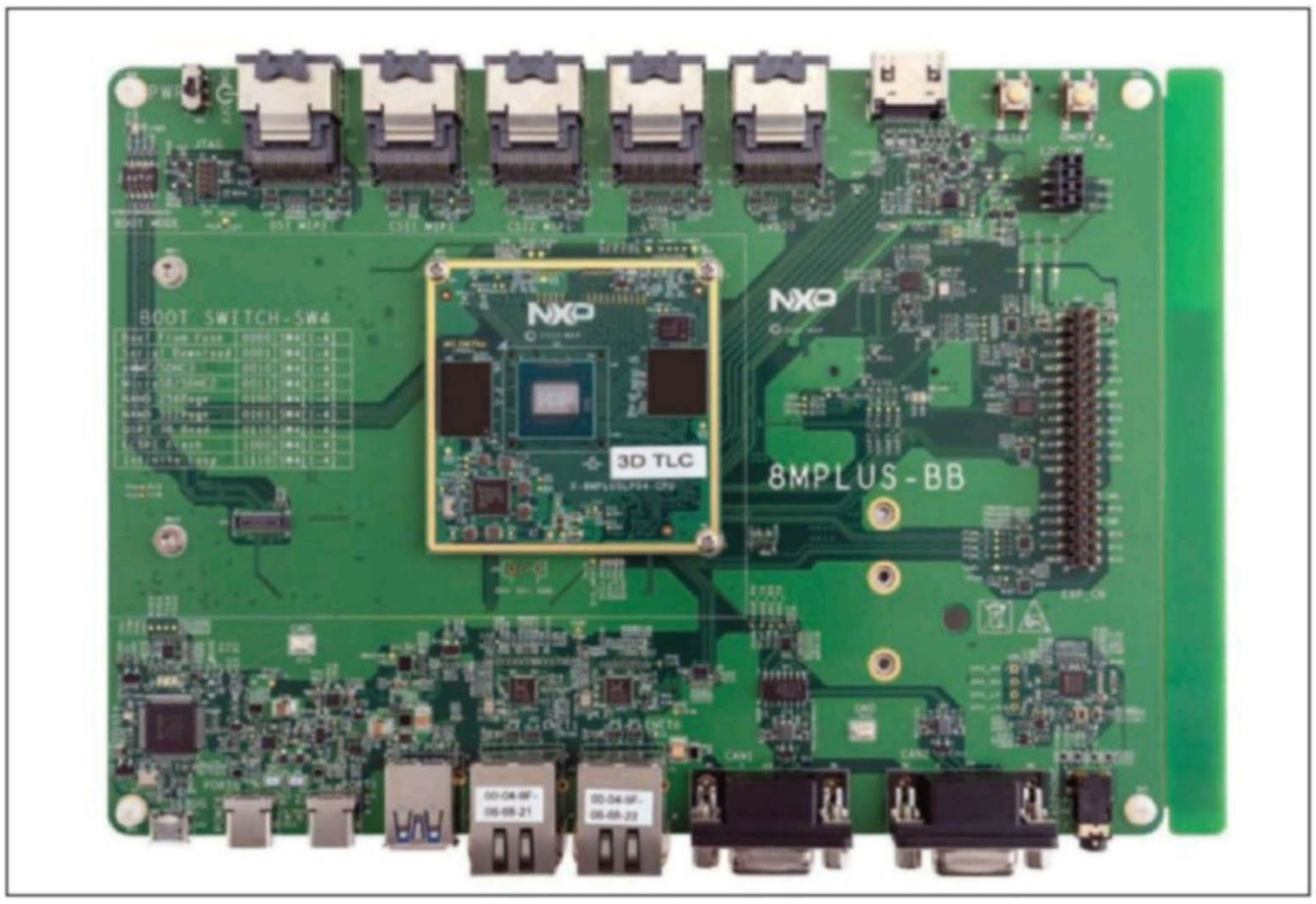

NXP
3D TLC
8MPLUS-BB

FIGURE 3
NXP Semiconductors' i.MX 8M Plus Evaluation Kit.

FIGURE 4
STMicroelectronics' STM-32N6 features the Neural-ART Accelerator, extending AI capabilities.

algorithms. These algorithms are often resource-intensive, requiring significant computational power and memory. To address this issue, several companies have developed specialized chipsets designed specifically for AI applications, making it easier for users to implement machine learning solutions without the need for extensive infrastructure.

One notable example is the i.MX 8M (**Figure 3**) Plus Applications Processor, which is part of a family of processors tailored for machine learning, advanced multimedia, and industrial automation. This powerful processor features either a quad or dual ARM Cortex A53 architecture, along with a neural processing unit (NPU) capable of performing up to 2.3 TOPS (tera operations per second). Additionally, it supports 32-bit DDR4 and LPDDR4 memory with speeds of up to 4.0GT/s, ensuring compatibility with cutting-edge technology. The robust capabilities of this chipset are essential for its built-in dual-image signal processor, which boasts a 12-megapixel resolution and supports two camera interfaces, alongside a 16 GFLOP GPU for enhanced graphics processing.

For applications that require a more compact solution, the STM32 Series Microcontrollers from STMicroelectronics offer a range of 32-bit ARM Cortex MCUs. These microcontrollers are designed to provide real-time capabilities in signal processing while maintaining low power consumption, making them ideal for various embedded applications. The STM32N6 (**Figure 4**) , in particular, is the first MCU in this series to incorporate the proprietary Neural-ART Accelerator neural processing unit (NPU). This innovative feature allows for efficient execution of machine learning tasks directly on the microcontroller. Users can explore various models available in repositories such as the STM32 model zoo, enabling them to customize the chip to effectively meet specific business needs.

As AI continues to evolve, its integration into various technologies will likely lead to even more innovative applications, transforming industries and enhancing the way we interact with machines. The future of AI is bright, and its potential to revolutionize processes and improve efficiency is boundless. What aspect of AI or its applications are you most interested in exploring further?

ROBOTICS

Key to the manufacturing side of automation is the field of robotics: broadly defined as the process of building machines to assist humans in tasks. In manufacturing, robotics can be used on assembly lines to automate assembly processes such as cutting, stamping, welding, and the mechanical assembly of parts. In other fields, robotics can serve to create autonomous vehicles or telemetry data collection units, such as the Curiosity Rover.

Boston Dynamics, one of the world leaders in consumer-end robotics, has released a new autonomous robot: Stretch (**Figure 5**). This wheeled, single-armed mobile manipulator robot is designed for loading and unloading packages up to 50 pounds in weight via an articulated suction cup surface for warehouse automation. Stretch allows for continuous workflow in repetitive tasks to reduce the need for employees to focus on such tedious tasks. With a mobile base with a footprint roughly the size of a standard pallet, and a real-time machine learning algorithm to guide its decision-making, Stretch has been designed as the future of warehouse automation.

With all robots, precise control of moving parts is essential for proper functioning. The precision and accuracy of a motor are essential parts of the finesse with which a robot can operate. One source of high accuracy, reliable

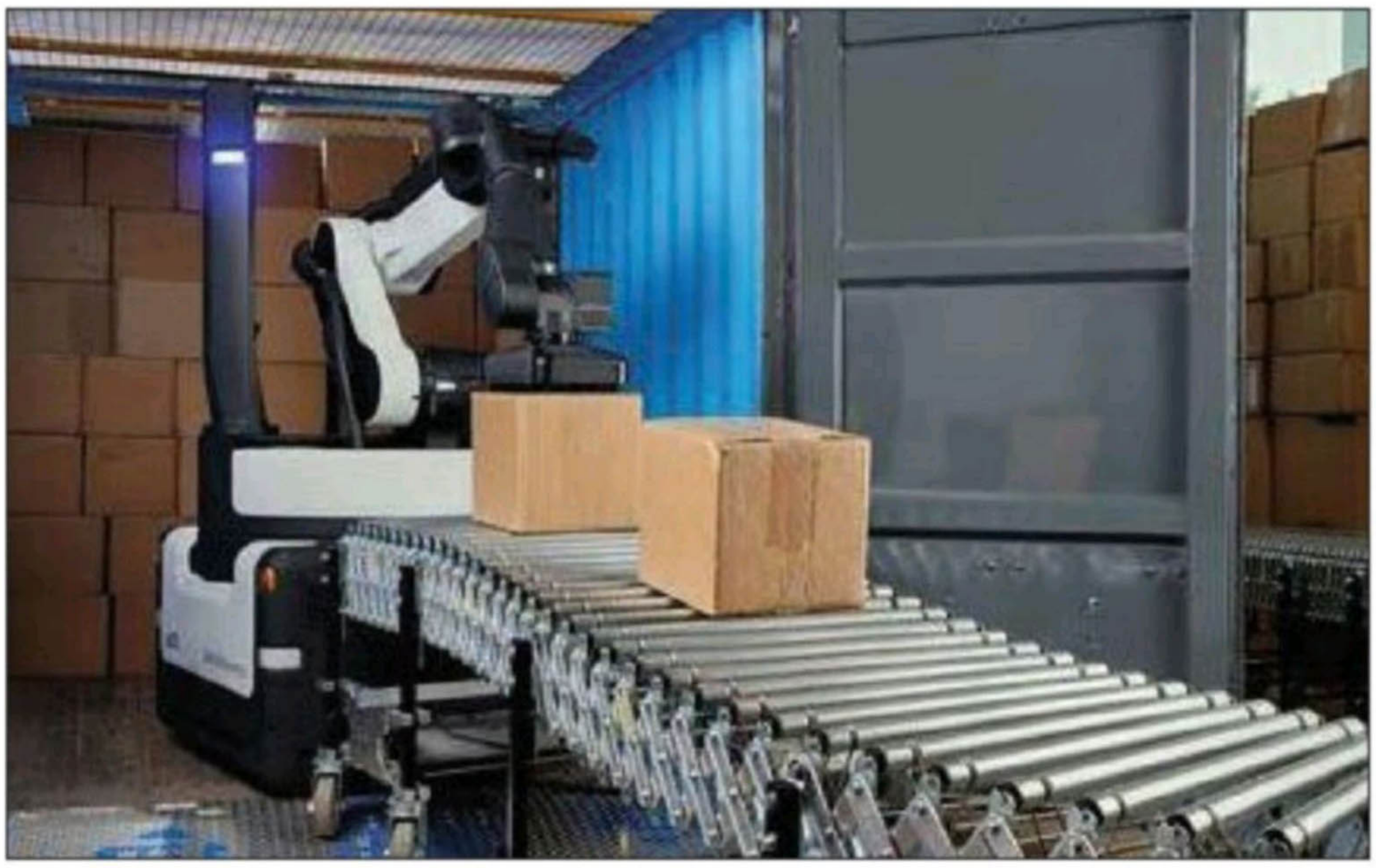


FIGURE 5
Boston Dynamics' Stretch is shown loading packages onto a conveyor belt.

motor drivers is STSPIN Motor Drivers, which offers drivers for brushed, brushless, and stepped motors. With a low voltage requirement of 1.8V and a standby current as low as 80nA, these motor drivers come with 10-year and 15-year longevity programs. With stepper motor drivers supporting 256 micro-steps per step, these drivers are capable of handling even high-precision applications with ease. These drivers are part of a development environment that includes the STM32 Nucleo development boards and the STM32 Nucleo expansion boards with standardized interconnections such as an Arduino Uno R3 connector and a morpho connector, for customization of advanced functionalities and are supported by STM032-based software modules.

Let's delve deeper into the companies that provide embedded boards for robotics in industrial automation.

NexCOBOT specializes in open robots and machines, integrating AI and functional safety into their embedded solutions (**Figure 6**). They aim to shape the future of smart manufacturing and enhance human-machine collaboration, making their products highly relevant in industrial automation. Their embedded boards are designed to be highly customizable, allowing developers to tailor them to specific needs. For example, their boards can be used to create robots that can perform tasks such as picking and placing objects, welding, and painting. NexCOBOT's commitment to open-source software and hardware makes their products attractive to a wide range of developers, fostering innovation and collaboration within the robotics community.

Softeq develops embedded software, drivers, and smart controls for industrial robots that handle various manufacturing tasks, including assembly and welding. Their solutions are designed to enhance efficiency and reliability in industrial settings. Softeq's expertise in embedded software development allows them to create solutions that are specifically tailored to the needs of industrial robots (**Figure 7**). They have a deep understanding of the challenges faced by manufacturers in automating their processes, and they use this knowledge to develop solutions that are both reliable and cost-effective. Softeq's embedded boards are often used in conjunction with other industrial automation components, such as sensors, actuators, and controllers, to create complete robotic systems.

ASRock Industrial offers high-performance embedded boards with flexible I/O options, longevity, and robust systems ideal for factory automation and autonomous vehicles. Their products are well-suited for a range of industrial applications. ASRock Industrial's embedded boards are known for their high performance and reliability, making them suitable for use in demanding industrial environments. They offer a wide range of boards with different processing capabilities, memory options, and I/O configurations, allowing developers to choose the best board for their specific needs. ASRock Industrial's boards are often used in applications where high performance and reliability are critical, such as in factory automation, robotics, and autonomous vehicles.

FIGURE 6
Smart robot solutions for manufacturing by NexCOBOT.

FIGURE 7
Softeq equips manufacturing companies with Smart Factory solutions designed to revamp assembly lines.

VersaLogic specializes in trusted embedded solutions for industrial automation applications. VersaLogic's embedded boards are designed to be rugged and reliable, making them suitable for use in harsh industrial environments (**Figure 8**). They offer a wide range of boards with different form factors, processing capabilities, and I/O configurations, allowing developers to choose the best board for their specific needs. VersaLogic's boards are often used in applications where reliability and long-term support are critical, such as in factory automation, robotics, and industrial control systems.

Toradex provides hardware and software solutions that are perfect for industrial Human-Machine Interface (HMI), controllers, and robots. Their products enable the creation of highly reliable systems quickly, making them a popular choice in the automation sector. Toradex's embedded boards are known for their ease of use and their comprehensive software support. They offer a wide range of boards with different processing capabilities, memory options, and I/O configurations, allowing developers to choose the best board for their specific needs. Toradex's boards are often used in applications where time-to-market is critical, such as in industrial automation, robotics, and medical devices.

These companies are at the forefront of providing essential components that enable the development of innovative robotic systems for industrial automation.

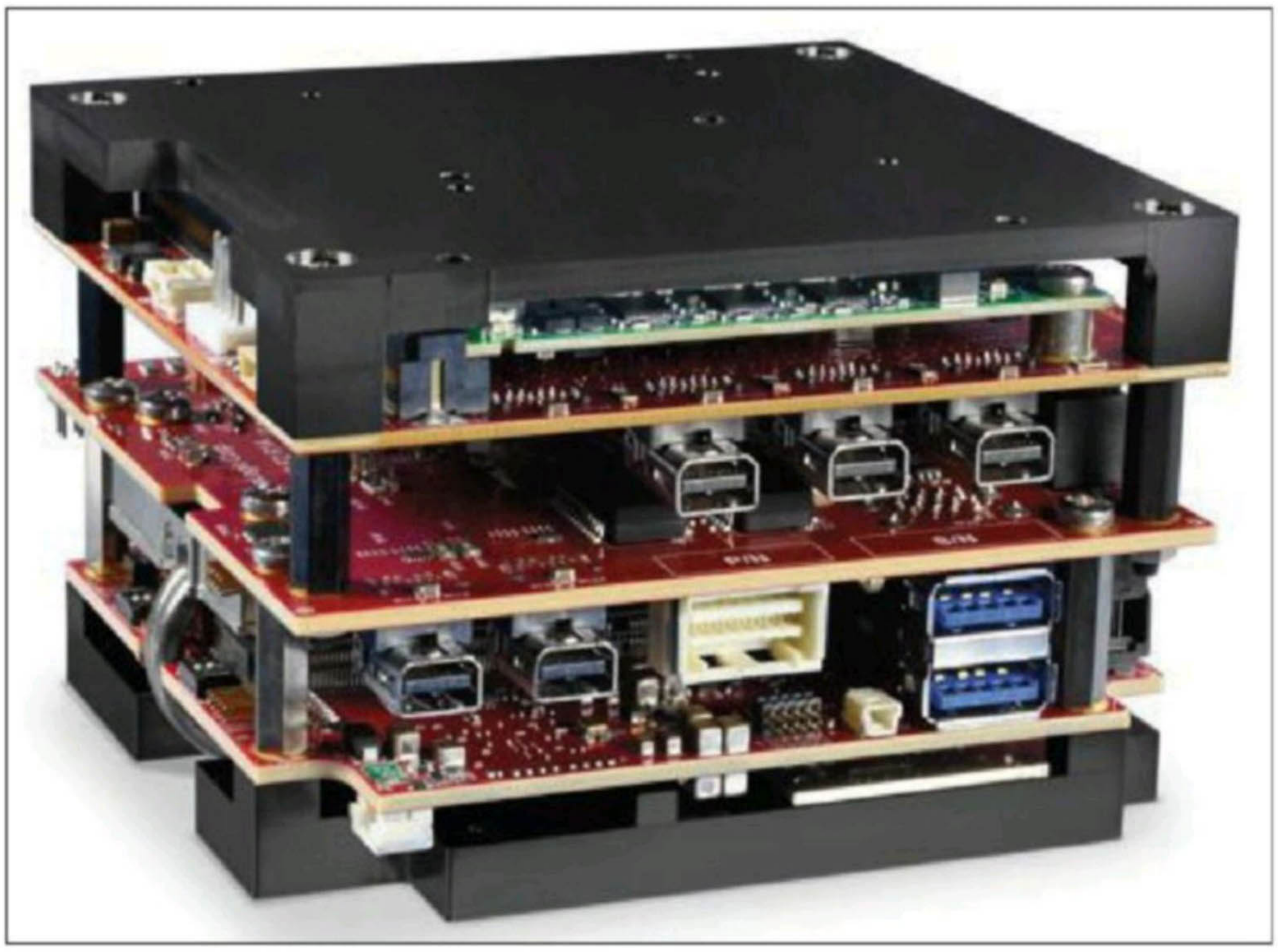


FIGURE 8
VersaLogic's NVIDIA TRX 2000 handles industrial temperature operations -40° to +85°C.

5G

The fifth generation of cellular communications network technology was launched in 2019. In the six years it has been operational, it has become the standard for telecommunications within the United States. By connecting geographically subdivided areas called cells to a base station through radio waves, this network enables access to the Internet and telecommunications network. Compared to the old 4G network, 5G has faster speeds, allowing for download speeds of up to 10Gbit/sec and wider bandwidth, which entails the connection of more devices. One of the first technologies to benefit from this new network was the Cellular V2X(CV2X) technology used by autonomous vehicles. This technology is composed of the V2V, V2P, V2I, and V2N networks as seen in **Figure 9**.

Swedish communications company Ericsson is currently looking into the benefits of 5G technology in industrial automation, and notes that the technology is specially tailored to IoT devices and can be used as Ultra Reliable Low Latency Communication, while also espousing its potential for massive machine-type communication for a large number of connected devices and sensors.

With 5G, it is possible to envision a future where factories are not just efficient but also flexible and adaptable. Imagine production lines that can be reconfigured on the fly to meet changing demands, or collaborative robots working alongside human workers, seamlessly sharing tasks and optimizing output.

The large bandwidth and low latency of 5G make communications among a large number of autonomous machines nearly instantaneous, and enables widespread communications among a multitude of machines and a central controller for real-time improvements to manufacturing efficiency by

Additional materials from the author are available at:
www.circuitcellar.com/article-materials

Resources

ASRock Industrial | www.asrockind.com

Boston Dynamics | www.bostondynamics.com

Ericsson | www.ericsson.com

Intel | www.intel.com

NexCOBOT | www.nexcobot.com

Softeq | www.softeq.com

STMicroelectronics | www.st.com

Texas Instruments | www.ti.com

Tordex | www.tordex.com

VersaLogic | www.versalogic.com

calibrating speeds, reducing waste, and even enabling AI driven predictive maintenance.

But as with any powerful technology, 5G comes with its own set of challenges. Ensuring robust security measures to protect sensitive data and prevent cyberattacks is paramount. Deploying 5G networks across large industrial sites can also be expensive, requiring careful planning and investment. And finally, there needs to be a skilled workforce with expertise in data analytics, cybersecurity, and robotics to fully harness the potential of 5G in industrial automation.

CONCLUSION

The past five years have seen huge leaps forward in automation. From how we approach tasks, how we communicate with each other and our equipment, to the capabilities of our manufacturing and industrial equipment, small changes have revolutionized the industry. With the advent of IoT and AI technology, as well as the improvements on previous technologies seen in these past years, whole new fields are open to automation, which were once thought to be forever safe from industrialization. With these benefits to our everyday and industrial lives comes new problems. AI in particular has endangered tens of thousands of jobs, similar to the advent of industrial manufacturing in the industrial revolution, leading to a negative perception of it among a large portion of the public. IoT has been shown to represent a credible threat to privacy and security, as it makes it easier than ever for data leaks to occur via spyware, or for network vulnerabilities to be found. With all this new technology, new branches of ethics must also be discussed, and we must conclude as a society what part in our lives do we want these new technologies to play.

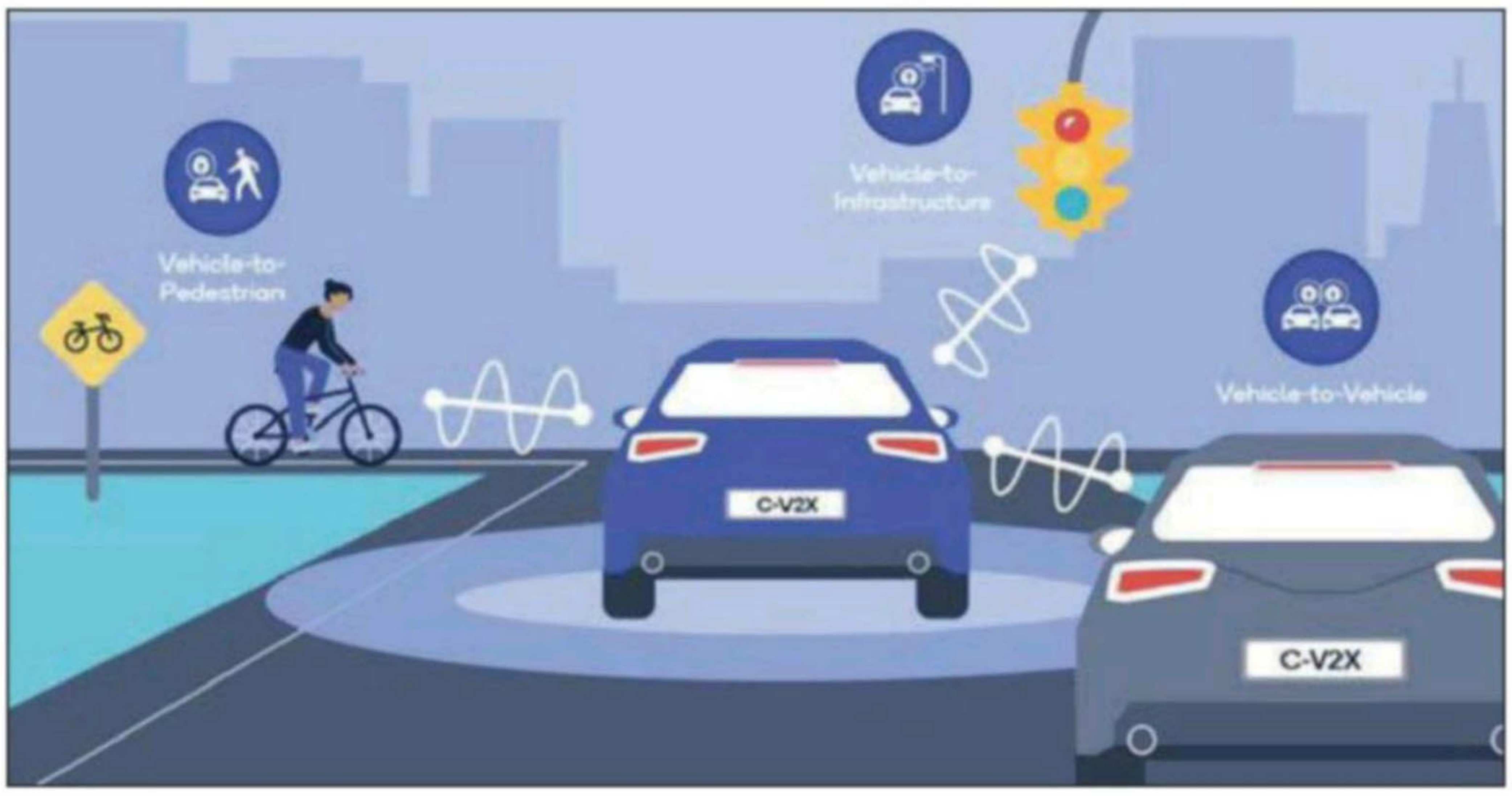


FIGURE 9
The V2X network allows autonomous vehicles to communicate with other vehicles, pedestrians, infrastructures, and remote networks through cellular technology.

MYiR 米尔电子 Make Your Idea Real

From $12.8 Customizable

SoM for Entry-level HMI & Embedded Devices with Video Capabilities

MYC-YT113i CPU Module

Allwinner T113-i

2×Cortex-A7@1.2GHz

Flexible Memory Configuration

1080P@60fps Video Decoding HiFi4 DSP

www.myirtech.com

Energy & Power

Commercial Display

Industrial Control

Smart Home
MYiR
PN:MYC-YT113i-4E512D-110-I
SN:WK2023062100400010
ALLWINNER T113-i

Datasheet: DSPs and DSCs

Smarter Algorithms and Beyond

By
Caleb Smith,
Editor-in-Chief

The world of digital signal processing (DSP) is rapidly evolving, with new products and advancements impacting every industry. Discover how engineers are leveraging AI, low-power solutions, and innovative software to push the boundaries of what's possible in audio, healthcare, and beyond.

Recent advancements in digital signal processing (DSP) technology have significantly transformed the landscape for professional engineers, enabling them to tackle increasingly complex challenges across various industries. One of the most notable trends is the integration of artificial intelligence (AI) and machine learning techniques into DSP systems. This integration allows for enhanced signal analysis and processing capabilities, enabling engineers to develop smarter algorithms that can adapt to changing conditions in real time. For instance, AI-driven DSPs can optimize audio and visual signals, improving clarity and reducing noise, which is particularly beneficial in applications such as telecommunications and multimedia.

Moreover, the demand for low-power DSP solutions has surged, driven by the proliferation of Internet of Things (IoT) devices. Engineers are now focusing on creating DSPs that maintain high performance while consuming minimal power. This is crucial for battery-operated devices, where energy efficiency is paramount. Recent innovations in low-power DSP architectures have made it possible to achieve significant processing capabilities without compromising on energy consumption, thus extending the operational life of portable devices.

In the realm of audio processing, new products have emerged that cater specifically to the needs of professional engineers. For example, advanced DSP audio matrices are now available, offering comprehensive supervision and system control for superior sound quality. These products enable engineers to manage complex audio environments with ease, ensuring optimal performance in settings ranging from concert halls to corporate boardrooms. Additionally, the introduction of embedded audio processing software has streamlined the design and debugging processes, allowing engineers to implement and tune audio signal flows in real time without extensive coding.

Another exciting development is the enhancement of DSP capabilities in healthcare imaging. Recent advancements have led to more sophisticated imaging techniques that improve diagnostic accuracy. Engineers are leveraging DSP technology to refine image quality and processing speed, which is critical in medical applications where timely and precise information can significantly impact patient outcomes.

Furthermore, the automotive industry is witnessing a surge in DSP applications, particularly in the development of advanced driver-assistance systems (ADAS). These systems rely on DSP technology to process data from various sensors, enabling features such as adaptive cruise control and collision avoidance. As vehicles become more connected and autonomous, the role of DSP in ensuring safety and efficiency will only grow.

In summary, the recent advancements in DSP technology and the introduction of innovative products are empowering professional engineers to push the boundaries of what is possible in their fields. By embracing these developments, engineers can enhance their designs, improve system performance, and ultimately contribute to the creation of smarter, more efficient technologies.

NEXT MONTH'S TOPIC: 32-bit ARM Microcontrollers Send related product announcements to editor@circuitcellar.com

A 16-bit Fixed-Point DSP

The ADSP-21992 from Analog Devices is a high-performance 16-bit fixed-point DSP designed for mixed-signal applications. It features a 160MHz core, enabling up to 160 MIPS of processing power, making it ideal for real-time signal processing tasks. The device includes an 8-channel, 14-bit, 20 MSPS ADC and an integrated CAN interface, enhancing its versatility in automotive and industrial applications. Its compact design and robust performance make the ADSP-21992 a standout choice for developers seeking efficient and powerful DSP solutions.

- Maximum clock frequency: 160MHz
- On-Chip RAM: 48K words
- Program memory size: 96kB
- Data RAM size: 32kB
- Integrated ADC 8-channel, 14-bit, 20 MSPS
- Interface SPI, SSP
- Package Mini-BGA-196
- External Non-Volatile Memory Supported
- PWM Generation Unit: 16-bit
- Programmable PWM Resolution: 12.5ns
- Power supply voltage: 3.3 V
- Operating temperature range: -40°C to +85°C
- Instruction set architecture: fixed point
- Development tools available
- Power consumption: Low
- Application areas: Automotive, Industrial, Audio Processing
- Performance efficiency: High MIPS/mW Ratio

Analog Devices
www.analog.com

A Mixed-Signal DSP Controller

The AK7709VQ is a cutting-edge DSP designed for high-performance audio applications. It features 7,000 MIPS processing capability and 2.75MB of integrated memory, enabling real-time high-resolution audio processing. This multicore DSP supports eight stereo and four monaural sampling rate converters, accommodating a wide range of sampling frequencies. Its advanced architecture is particularly suited for automotive and consumer electronics, making it a standout choice for engineers seeking efficiency and versatility in audio processing solutions.

- Processing capability of 7,000 MIPS
- Integrated memory of 2.75MB
- Supports eight stereo sampling rate converters
- Supports four monaural sampling rate converters
- Sampling frequency support up to 192kHz
- Low power consumption design
- High-resolution audio processing capabilities
- Multicore architecture for enhanced performance
- Built-in digital filters for audio enhancement
- Flexible audio interface options
- Compatible with various audio formats
- Integrated noise reduction features
- Supports I^2S and TDM audio data formats
- Designed for automotive applications
- Robust error correction mechanisms
- Compact package size
- High reliability for long-term use

Asahi Kasei Microdevices
www.akm.com

High-Performance Optical DSP

The Aquila Coherent-Lite DSP, a revolutionary product from Marvell Technology, is a high-performance optical DSP designed for data center applications. This DSP, optimized for 1.6Tbps coherent optical transceiver modules operating in the O-band, delivers low power consumption and latency, spanning distances from 2km to 20km. Featuring advanced data handling capabilities and 100 Gbps/lane PAM4 host electrical interfaces alongside 400Gbps/lane 16-QAM optical interfaces, the Aquila DSP seamlessly integrates into existing network infrastructures. Its compact design and innovative O-band coherent architecture enhance cost efficiency and scalability.

- Processor type: Coherent optical DSP
- Maximum data rate: 1.6Tbps
- Operating wavelength: O-band
- Power consumption: Low power design
- Latency: Low latency performance
- Distance: Range 2km to 20km
- Host electrical interface: 100Gbps/lane PAM4
- Optical interface: 400 Gbps/lane 16-QAM
- Form factor: Pluggable module
- Applications Data center interconnects, AI data centers
- Scalability supports future expansion
- Designed for cost-effective deployment
- Supports Ethernet applications
- Integration compatible API and software suite
- Signal processing: Advanced coherent signal processing capabilities
- Unique Architecture: Innovative O-band coherent architecture

Marvell Technology
www.marvell.com

DATASHEET URLS:

Analog Devices: https://www.analog.com/media/en/technical-documentation/data-sheets/ADSP-21992.pdf

Asahi Kasei Microdevices: https://www.akm.com/content/dam/documents/products/audio-voice-processor/ak7709vq/ak7709vq-en-datasheet.pdf

Marvell: https://www.marvell.com/content/dam/marvell/en/public-collateral/dsp/marvell-aquila-coherent-lite-dsp-product-brief.pdf

Datasheet:
DSPs and DSCs

MICROCHIP
dsPIC33A DSCs

Cutting-Edge DSCs

The dsPIC33A series from Microchip is a cutting-edge digital signal controller designed for high-performance applications. It features a 32-bit CPU operating at 200MHz, optimized for real-time processing and advanced mathematical computations. The architecture includes a DSP engine with 72-bit accumulators and a DP-FPU for accelerated floating-point operations, ideal for complex algorithms. With low power consumption and robust functional safety features, the dsPIC33A is designed for engineers developing energy-efficient and safety-critical systems.

- 32-bit CPU
- 200MHz operation
- DSP engine with 72-bit accumulators
- DP-FPU for accelerated math operations
- Comprehensive instruction set for optimized speed
- Non-paged linear Data/Flash memory
- Integrated peripherals for enhanced functionality
- Low power consumption
- Advanced functional safety features
- Multiple communication interfaces (I^{2}C, SPI, UART)
- 16-bit and 32-bit data types support
- Up to 128KB Flash memory
- Up to 16KB RAM
- 10-bit ADC with multiple channels
- PWM outputs for motor control
- Capacitive touch sensing capabilities
- Development board support for rapid prototyping

Microchip Technology
www.microchip.com

MaxLinear announces next generation 200G/lane PAM4 DSP
MAXLINEAR

Next-Generation Technology

The Rushmore family of 200G/lane PAM4 DSPs from MaxLinear represents a significant advancement in high-speed data processing technology. Designed to meet the increasing demands for data traffic, these DSPs offer exceptional performance, power efficiency, and density. They are particularly suited for applications in data centers and high-performance computing environments. With their ability to handle complex signal processing tasks, the Rushmore DSPs stand out due to their innovative architecture and robust features, making them invaluable for engineers working on next-generation communication systems.

- Supports 200G/lane PAM4 signaling
- 4th generation SERDES technology
- High power efficiency for reduced operational costs
- Optimized for data center applications
- Integrated signal conditioning capabilities
- Low latency for real-time processing
- Scalable architecture
- Enhanced thermal performance
- Supports multichannel configurations
- Designed for high-density integration
- Robust error correction mechanisms
- Flexible I/O options for diverse connectivity
- Compatible with industry-standard protocols
- High data throughput capabilities
- Integrated clock recovery features
- Supports advanced modulation schemes
- Designed for low power consumption

MaxLinear
www.maxlinear.com

NXP
SAF9000

An Automotive Audio DSP

The SAF9100 from NXP is a state-of-the-art automotive audio DSP designed for the software-defined car, enabling advanced infotainment systems. It features integrated AI and machine learning accelerators, enhancing audio processing capabilities such as voice recognition and noise cancellation. The SAF9100 stands out with its support for multiple audio standards and its ability to process complex audio algorithms in real time. This makes it an essential component for engineers looking to develop next-generation in-car audio experiences that prioritize user interaction and sound quality.

- Automotive audio one-chip solution
- Integrated AI and ML accelerators
- Voice pitch and accent recognition
- Advanced noise cancellation features
- Software-defined radio option
- Up to five integrated tuners for global broadcast
- Multi-standard analog and digital radio processing
- High-performance audio processing capabilities
- Real-time audio algorithm execution
- Scalable architecture
- Low power consumption
- Robust functional safety features
- Multiple communication interfaces (I^{2}C, SPI, UART)
- Support for various audio formats
- Enhanced user interaction capabilities
- Development support for rapid prototyping
- Compact design for easy integration

NXP Semiconductors
www.nxp.com

DATASHEET URLS:

Microchip: https://ww1.microchip.com/downloads/aemDocuments/documents/MCU16/ProductDocuments/DataSheets/dsPIC33AK128MC106-Family-Data-Sheet-DS70005539.pdf

MaxLinear: https://www.maxlinear.com/products/infrastructure/wired-infrastructure/data-center-connectivity/pam4-dsps

NXP Semiconductors: https://www.nxp.com/products/SAF9100

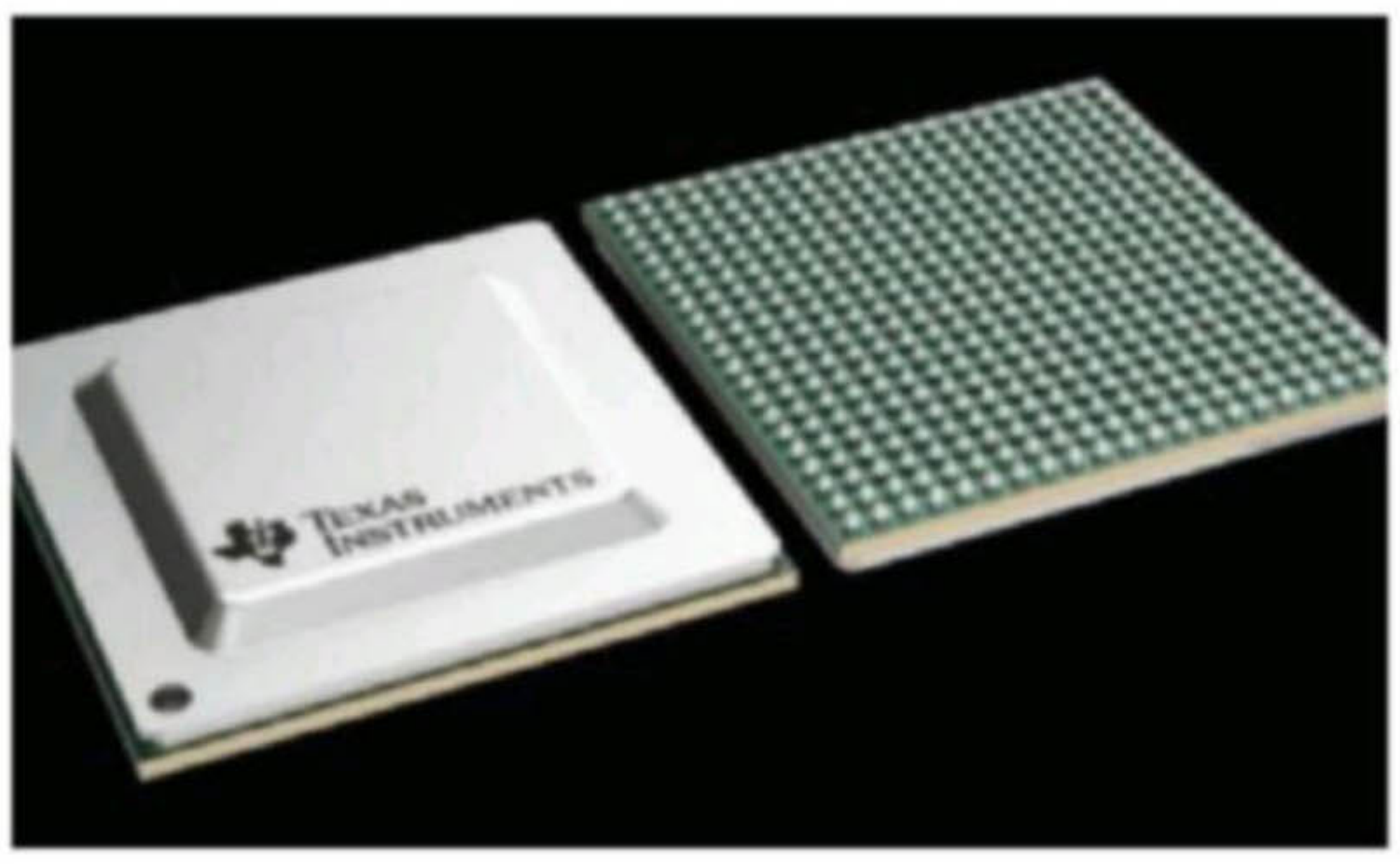
EZAIRO 8300

Designed for Hearables

The Ezairo 8300 is an advanced audio processor designed specifically for hearing aids and hearables, featuring six programmable or semi-programmable processing cores that provide exceptional parallelism and flexibility. Its architecture allows for high-quality audio processing while maintaining ultra-low power consumption, making it ideal for battery-operated devices. The processor includes four ADCs with signal detection modes and two direct digital output drivers, ensuring superior audio fidelity. These features make the Ezairo 8300 a standout choice for engineers focused on developing cutting-edge audiology solutions.

- Six programmable or semi-programmable processing cores
- Four ADCs with signal detection mode
- Two direct digital output drivers
- Ultra-low power consumption
- High-quality audio processing
- Supports various audio processing algorithms
- Programmable gain control
- Integrated noise reduction features
- Flexible I/O configurations
- High dynamic range
- Low total harmonic distortion
- Supports multiple audio formats
- Real-time processing capabilities
- Compact design for easy integration
- Enhanced signal-to-noise ratio
- Robust performance in challenging environments
- Open-programmable architecture
- Designed for wireless-enabled applications

Onsemi
www.onsemi.com



Automotive Audio Applications

The AM2754-Q1 from Texas Instruments is a powerful 80 GFLOPS DSP microcontroller designed specifically for automotive audio applications. It features a quad-core ARM Cortex-R5F architecture, that enables high-performance processing with each core running up to 1GHz. This microcontroller supports advanced audio processing capabilities, including spatial audio and active noise cancellation, making it ideal for enhancing in-vehicle audio experiences. With 10.75MB of SRAM, it provides ample memory for complex algorithms, ensuring seamless operation in demanding automotive environments.

- Quad-core ARM Cortex-R5F CPU
- Each core runs up to 1GHz
- 80 GFLOPS processing capability
- 10.75MB SRAM
- 32KB I-Cache with 64-bit ECC per core
- Integrated DSP for audio processing
- Supports spatial audio features
- Active noise cancellation capability
- Designed for automotive applications
- High reliability for automotive standards
- Low power consumption
- Flexible I/O options
- Advanced security features
- Real-time load diagnostics
- Current sensing capabilities
- Supports various audio interfaces
- Compatible with automotive audio frameworks
- Enhanced thermal performance

Texas Instruments
www.ti.com

xcore.ai
know more.
do more.

High-Performance Audio Solution

The xcore.ai platform from XMOS is a cutting-edge solution designed for high-performance audio and voice applications, integrating advanced DSP capabilities with real-time processing. It features 16 logical cores across two multi-threaded tiles, enabling efficient execution of complex algorithms. With its predictable programmability and low latency, xcore.ai is ideal for edge-AI applications, allowing for local data processing and immediate action. This platform stands out due to its flexibility, scalability, and robust performance, making it a valuable tool for engineers in developing intelligent IoT solutions.

- Up to 3200 MIPS of compute power
- 16 logical cores divided into 2 multi-threaded tiles with 512kB of SRAM
- Supports integer and floating-point ALUs
- 8 FIR taps (32-bit) per clock cycle
- Integrated AI acceleration capabilities
- Low eBOM (electronic Bill of Materials) solution
- Designed for real-time performance in audio applications
- Supports local data processing for edge-AI
- Vector unit for enhanced processing efficiency
- Compatible with Audio Weaver for audio processing
- Multi-threaded architecture for high determinism
- Low power consumption for embedded applications
- Flexible I/O options
- Ideal for intelligent IoT applications

XMOS
www.xmos.ai

DATASHEET URLS:

Onsemi: https://www.onsemi.com/download/data-sheet/pdf/e8310-d.pdf

Texas Instruments: https://www.ti.com/lit/gpn/AM2754-Q1

XMOS: https://www.xmos.com/documentation/XM-014034-PC/html/rst/XU316-1024-QF60A.html

PICKING UP MIXED SIGNALS

An Air Quality Monitor with a Blues Satellite Link Module

Brian had no sooner finished the review of Blues' LoRa Notecard when the new Starnote module was introduced. This module allows an IoT device to link to a geosynchronous satellite, for situations in which neither WiFi nor Cellular data links are operational. This article describes a project using the Starnote module.

SIGLENT
Stop
f = 13.00000Hz
C1
Utility
Display
Acquire
Trigger
Cursors
Meas
Analysis
UART
Time
RX
RX Err
TX
TX Err
Timebase
Trigger
C1 DC
16:22:57
2024/8/8

By
Brian Millier

In *Circuit Cellar* issue #410 (February 2025), I reviewed the Notecard LoRa Starter kit by Blues. This is a LoRa-based communication card along with a pre-configured LoRa Ethernet/WiFi gateway. Blues makes a series of Notecards that connect to their Notehub cloud service via WiFi, WiFi/Cellular or LoRa. Blues recently introduced Starnote, which is a module that also connects to the same Blues Notehub, however it does so via a satellite link.

Just as I was finishing work on the LoRa Notecard review, I was pleased to receive the new Starnote Starter kit from Blues. It made sense to investigate this module quickly, since all Blues Notecards work very similarly, and I had spent time learning how to use the Notecard LoRa for that review. I didn't have a specific use case for the Notecard LoRa, as I don't have sensors and such spread apart over long distances. However, I could see a use for the satellite-based Starnote and decided to design a project around it for this article.

As soon as you move away from WiFi or Bluetooth communication modes, you must consider the availability and cost of the data plan needed to implement your communication link. This can be challenging for an independent user such as myself. What Blues has done is integrate the data plan into the purchase price of the Notecard itself. In the case of their cellular-based notecards, this is in the form of a 10-year, 500MB cellular data plan, included in the Notecard price.

In the case of Starnote, satellite data traffic is much more expensive, so the data plan included with the Starnote is for 18KB of data (with no time limit). Blues Notecards generally use small data packets for transferring data to the cloud (and back) and when using either a Starnote or a Notecard LoRa module, an even more compact packet format is available and must be used. A meaningful example of this would be 50-byte packets (enough for several sensors) sent once per day. With the free 18KB data allotment, this would allow the Starnote module to operate for a year. Satellite usage figures are now shown in the Notehub dashboard. When you run out of your bundled data, you're charged at $0.75/KB, and that can be done via credit card.

STARNOTE'S SATELLITE PARTNER

With all the low-earth-orbit (LEO) Starlink satellites that Elon Musk has put into orbit, it would be natural to think that the Starnote modules might make use of those. While some high-end cellphones can already use Starlink for SMS messages, this is not the satellite constellation that Starnote uses. Instead, it uses the Skylo satellites, which are a constellation of much fewer satellites, orbiting in a geosynchronous orbit high above the equator. Since the satellites are geosynchronous, the Starnote module needn't track the satellite (by re-positioning its antenna) nor wait until one of them comes into view. That's assuming you have a clear view in the sky in the direction of the equator. Currently, the Skylo satellite coverage includes North America and Europe, but parts of South America, Australia, Japan, and southern Asia are in the works. The coverage map can be viewed at: www.skylo.tech/resources/geographical-coverage

So, what kind of antenna do we need to access these satellites reliably? You certainly don't need a "dish" antenna as was needed for satellite TV (Echostar, DirectTV, etc). Nor does the antenna need to be precisely aimed as we had to do with those TV satellite dishes. The Starnote starter kit comes with an antenna which they claim is sufficient to link up to the Skylo satellite constellation. **Figure 1** shows that antenna made by Quectel. As you can see, it's quite small and is a "patch" antenna with an adhesive backing. The tiny U.FL connector can be tricky to install—be careful you don't break it. I have not been able to find a replacement source for this antenna, at my usual distributors. I mentioned to Blues that it might be a good idea to add that to their list of accessories, but it wasn't on their site at the time of writing.

To contact the satellite, the Starnote module must be connected to this antenna and a clear southerly view of the sky must be available (in the northern hemisphere). Blues states that the antenna must be placed outdoors. At the rear of my house, there are a lot of trees blocking most of the southern view. My house lot borders a large lake however, which presents an unobstructed view to the south, so I first tested Starnote down by the lake. When I was able to successfully connect to the Skylo satellite, my confidence rose and I next tried it from the deck at the back of my house, which had its southern view substantially blocked by trees. I was still able to connect from there, so I then tried from inside the house through a south-facing picture window with roughly the same obstruction from the trees mentioned earlier. I was still able to connect from this location. For interest's sake, I live almost exactly on the 45° Latitude line. In the live demonstration that Blues held (via Zoom meeting), they used an outdoor cellphone antenna to connect with the satellite, since they had to shoot that video indoors. I tried this, unsuccessfully, with a 6dBi whip antenna designed for 4G LTE. On the Blues forum, others have reported trying similar antennas with no success. Blues' response is that it's likely the coax cable needed to reach the outdoor location, combined with the necessary U.FL to SMA adapter cable, likely attenuate the signal enough to make this arrangement unusable.

NOTECARD FUNCTIONALITY

Before describing the way that the Starnote module works, it's important to mention one important technical fact. Unlike the other Blues Notecards (WiFi, cellular, combo or LoRa), the Starnote module requires one of those other Notecards to be present to act as the master device. Part of the reasoning here is that the Starnote is sending messages to a satellite, and this is a much more expensive way to handle the traffic than either a WiFi link, or even a cellular link. Therefore, the design philosophy behind the Starnote module is that it will normally serve as a fail-back device. Your messages would go to a Notecard, which would attempt to send them via WiFi or cellular (or maybe both, if you use a combination WiFi/cellular Notecard). Should that fail, the Notecard will then "fire up" the Starnote module and send the message via satellite. Assuming you have a clear view of the fixed (geosynchronous) satellite serving your area, there is no reason why that message should fail to get through.

Since one of the Notecard modules is required to use Starnote, I'll quickly cover their operation. A more in-depth explanation can be found in my article in *Circuit Cellar* #410. **Figure 2** shows the interconnection of the various modules needed to implement the Starnote-based IoT device that is the subject of this article.

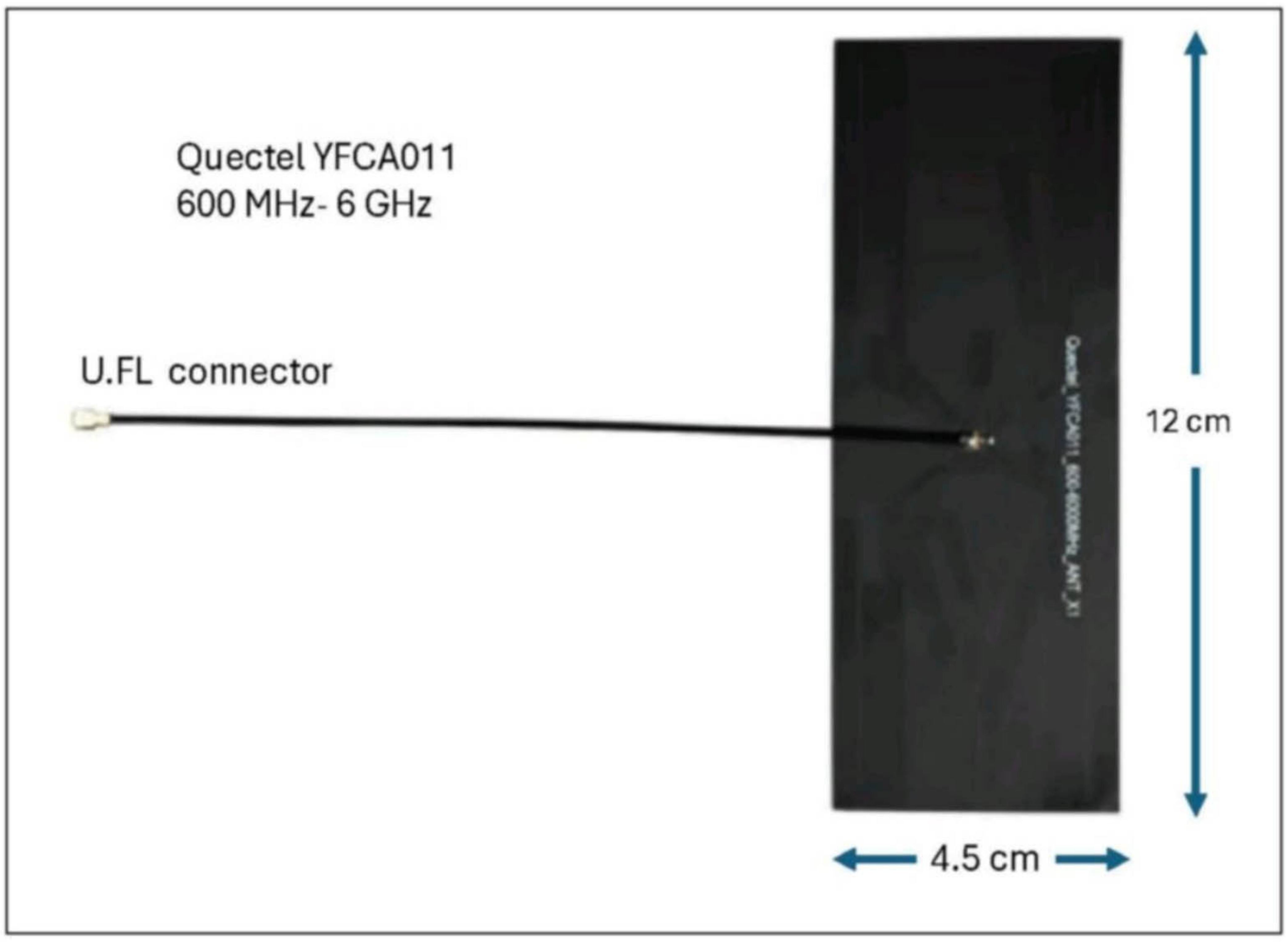


FIGURE 1
This is the Quectel Satellite Antenna supplied with the Starnote Starter Kit. The U.FL connector is fragile so be careful installing it.

The RF-enabled MCUs and SOCs that I have used in the past, for custom IoT devices in my home, were all complete units. The same MCU that handled the RF communication protocol stack also executed the user's program. My experience was with ESP32s and Particle modules, for the most part. In contrast, all the Notecards are described as "data pumps." That is, they handle one or more of the RF protocols needed to establish wireless communication with a central controller/gateway. They don't execute the user program functions (i.e., reading sensors, driving actuators, etc.). Those tasks must be handled by a separate, user-supplied MCU, which communicates this information, bidirectionally, to the Notecard via UART, I²C or USB. The designer is free to choose whatever MCU he/she is most comfortable with—as long as it has a UART or an I²C port.

Another distinction is that the Blues Notecard's firmware is specifically designed to seamlessly connect to the Blues cloud server. Any necessary authentication information needed by the Blues Notehub cloud service is pre-provisioned in the Notecard itself.

The Blues Notehub cloud server is structured to act only as a temporary "middleman" for the data to/from your IoT device(s). This cloud server is set up to allow "routings" to various other commercial cloud providers, which offer a full suite of services.

You might consider that the need for two MCUs is a disadvantage—especially if you are doing a low-power application. However, the Notecards use very-efficient MCUs. According to Blues documentation, the various Notecards draw ~8µA when in idle mode (waiting for commands from the host MCU).

For all Notecards, messages containing sensor values, and more are sent on to the Notehub server, and are called "Notes." They resemble the MQTT "publish" or "subscribe" format, in that each Note will include one or more user-defined descriptions of variable(s) plus a value for that variable(s). These Note definitions, in JSON format, are not something that you can spontaneously define and then send out to Notehub. First you must define a template for each Note, which describes the exact format of the JSON structure used for that Note. For example:

```
"req":"note.template","file":"data.qo",
"port":1,"format":"compact","body":
{"temp":14.1,"humid":14.1}}
```

Here, we are making a **req**uest to define a note template named "data.qo". The "qo" indicates that the data is outgoing, and "data" can be whatever filename you wish to use. For now, ignore `"port":1`, as I will describe it later. The format is defined as `"compact"`. For satellite use, with Starnote, these notes should be as small as possible. After `"body"`, you define the variable(s) that you wish to send within that Note. Here again, you can name these according to whatever makes sense for your application. The two variables shown are "temp" and "humid," which are self-explanatory. However, they are each followed by 14.1 and this bears explanation.

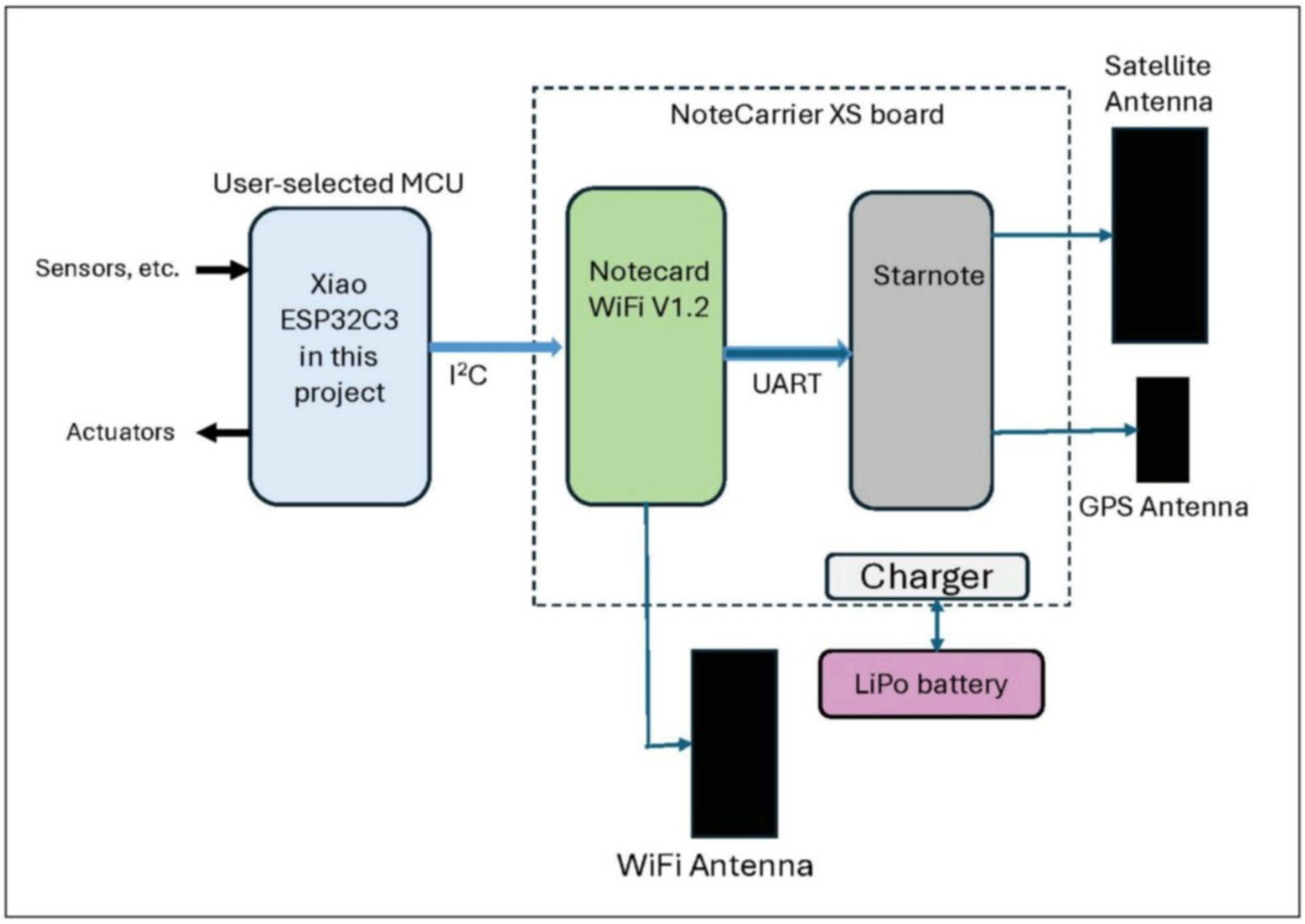


FIGURE 2
This is a block diagram for a generic project using the Starnote module, a WiFi Notecard, and the user's host MCU. Both the Starnote and the Notecard are mounted on a Notecarrier XS carrier board.

Variables can be expressed as integers, floats, Strings, or Boolean—it's your choice. While the 14.1 looks a bit like the way that the "printf" format specification works, it's not. To see a table of this numeric format definition, see the Notecard documentation under the "Low Bandwidth Design" section.

The section of the template `"port":1` is needed for the StarNote (satellite) modules. It's important to minimize the size of each Note because of the cost of sending each packet in the case of a satellite connection. Each Note template must have its own unique port number—in the range of 1-100. I suspect that both the Notecard(s) and Notehub will have a list of your project's templates, defined by port number. When Notes are sent/received, only this port number would be sent—the template structure could be looked up at the other end, using this port number to index a template list. With that template, it's possible to properly interpret the data, parsing it into the correct numeric or String format.

The procedure for sending notes is well-laid out in Blues' Starnote Quickstart guide. A link to this guide can be found on the "Dev Kits/Starnote Starter Kit" webpage. Blues has provided a lot of good documentation on their products, most of it readily available on their website. One innovation is that when navigating through these Quickstart guides, there are many points at which the right section of the screen will be occupied by a black dashboard prompting you to connect to your Notecard via a USB-Serial link. Clicking on this button will open the Notecard's In-Browser Terminal and will connect to any Notecard that is plugged into your PC via a USB cable. This can be seen in **Figure 3**. This connection is a USB COM port connection, and the COM port number is determined automatically by the In-Browser Terminal. Window's Device Manager will show the Notecard as a Silicon Labs' CP210x USB to UART bridge. Note: the In-Browser Terminal will only work on browsers that feature the web-serial API (i.e., MS Edge and Chrome).

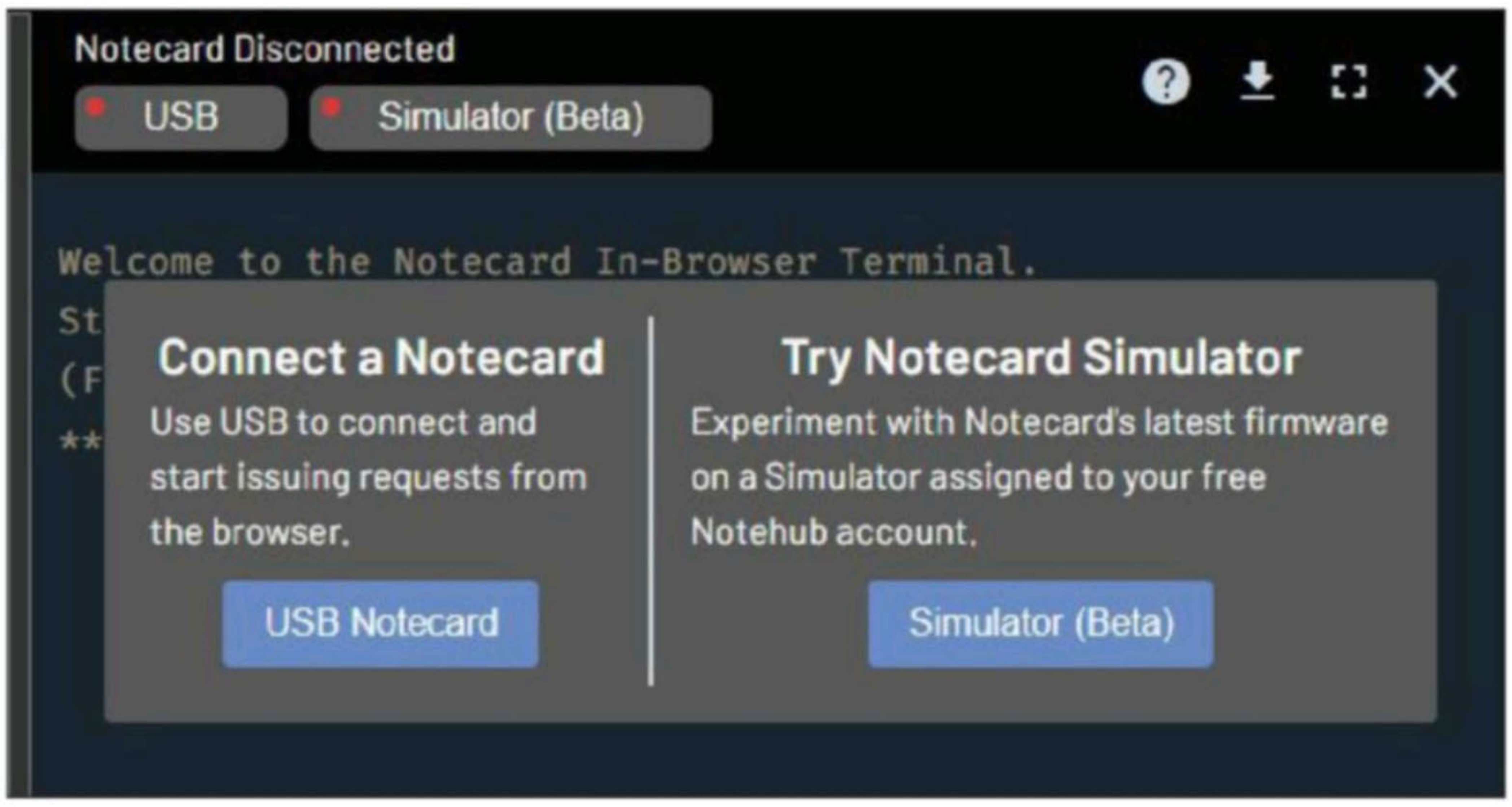
In this article, I'll describe a project that I built to measure PM2.5, CO_2, Temperature, and Humidity. These four values, along with the LiPo battery voltage are sent to Notehub as a single Note. While these sensors are normally used indoors, I had another alternate use in mind. I experienced a situation last summer in which a wildfire burned through a village nearby. I realized that everyone gets evacuated quickly and it can be weeks before you know whether or not your own house was destroyed. I thought that it would be useful to be able to place the monitor that I describe in this article, outdoors, prior to an evacuation. The readings for PM2.5 and CO_2 would definitely increase if the fire approached the property, and the temperature would rise significantly if a fire invaded the property. Therefore, the monitor would be a good source of information on the status of the house, Granted, a camera could provide an image of the house, removing any doubt, but the Starnote satellite link would not have enough data allocation to transport even a compressed image.

In such wildfires, the electricity to affected areas is cut off, for the safety of the firefighters. While my WiFi router has battery backup, it only lasts a day or so without human intervention. This monitor would normally use WiFi as its transport mode, but if that fails, the Starnote would send the sensor data via satellite. The monitor itself has a 5000mAh LiPo battery and its power consumption is discussed in a later section.

VINDRIKTNING PM2.5 SENSOR

One of the common measurements involved in air quality these days is PM2.5. This stands for particulate matter in the 2.5μm size range: much smaller than the eye can discern. PM2.5 sensors generally measure this by moving a small quantity of the surrounding air into a chamber. A narrow beam of light is sent through this air and a photosensor is mounted at right angles to this light beam. Due to the geometry of this arrangement, the photosensor will not see the direct light from this beam, but light that hits the PM2.5 particles will be reflected in all directions and some of these reflections will be picked up by the photosensor. The quantity of light picked up this way will be proportional to the concentration of PM2.5 particles in the air.

In the past, such PM2.5 sensors were expensive, since they were used for scientific purposes and thus had to be well-calibrated. In recent years, these sensors have become



```
Welcome to the Notecard In-Browser Terminal.
St
(F
**
```

Connect a Notecard

Use USB to connect and start issuing requests from the browser.

Try Notecard Simulator

Experiment with Notecard's latest firmware on a Simulator assigned to your free Notehub account.

FIGURE 3
This is a screenshot of the Blues In-Browser Terminal that can be used to communicate with Notecards, Starnote, and more via USB.

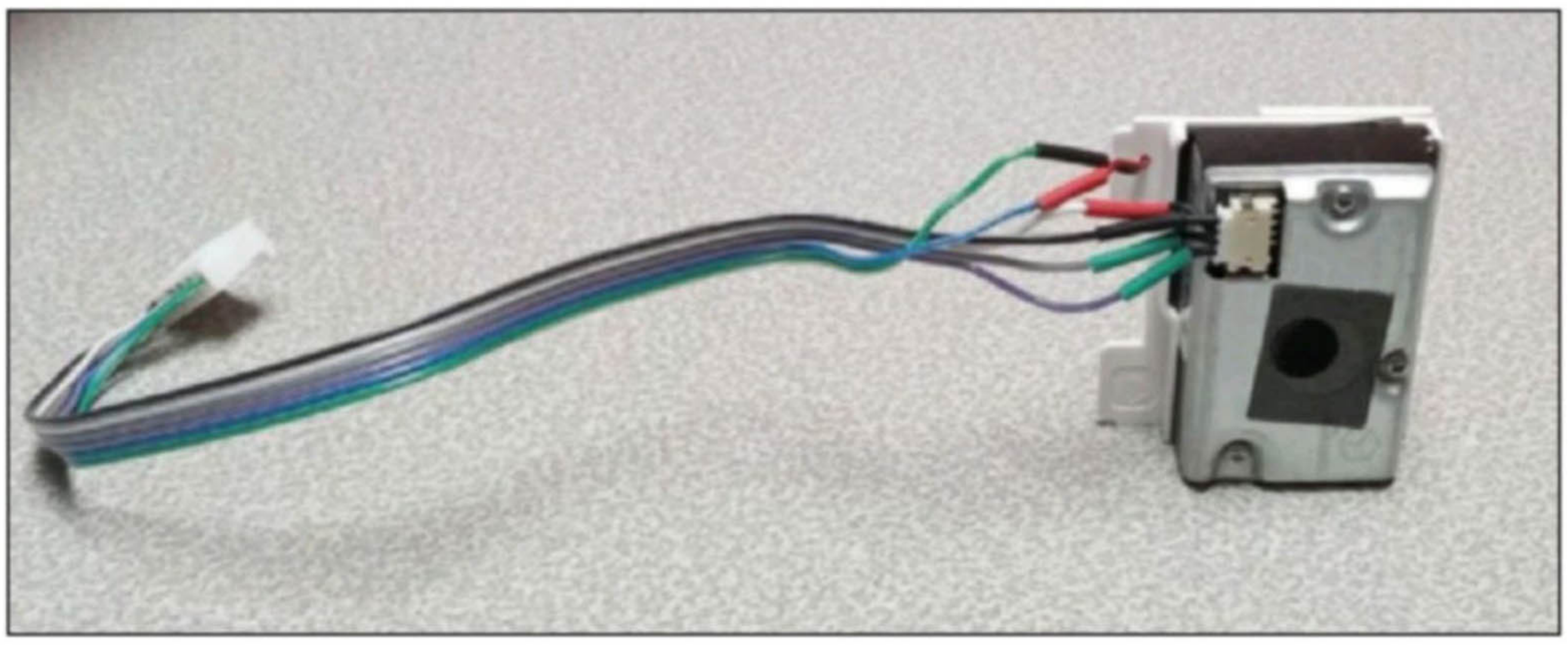


FIGURE 4
This is the PM1006K PM2.5 air quality sensor that I used. It was removed from a $10 IKEA Vindriktning Air quality monitor.

more of a commodity item, as they are found in air purifiers and other air-handling devices. Several large companies such as Sensirion, Panasonic, and Honeywell make such devices in the $40-$150 range, some of which were in stock at my preferred distributors.

Thanks to Google, I stumbled upon another source of these sensors. I can't take credit for this discovery, but it turns out that IKEA makes a $10 air quality monitor, the Vindriktning (PN 605.159.11). Others have torn the Vindriktning apart and found out that it contains a PM1006K PM2.5 sensor, made by Cubic. Along with the PM1006K sensor, the Vindriktning contains a small control board containing an 8-pin MCU and a tricolor LED. It's a slick design—the MCU is unlabeled, but six lines are needed to control the LEDs, the fan, and the PM1006K, so every pin on the MCU is being used. **Figure 4** shows a photo I took of this sensor. It operates on the same principle as I've already outlined and contains a small 5V fan to pull the air in. The main connector on this sensor is a 4-pin 1.5mm connector, which I didn't have, but it was easy enough to cut the cable that connected it to the Vindriktning control board, and splice in a 6-wire ribbon cable with a 0.1" plug. This cable also powers the fan.

The PM1006K sensor uses a UART interface, and the commands needed to operate it are contained in the PM1006K datasheet, which can be found at: https://en.gassensor.com.cn/Product_files/Specifications/LED%20Particle%20Sensor%20PM1006K%20Specification.pdf

The PM1006K UART's Rx, Tx signals are at 5V levels but the ESP32C3 MCU that I was using as the controller, uses 3.3V logic. I reduced the PM1006K's Tx signal down to 3V, using two resistors. The PM1006K's datasheet doesn't state what its Rx logic levels must be to work properly. I am used to 5V devices being able to properly accept 3.3V logic levels. However, I added a logic level translator to the PM1006K's Rx line.

The PM1006K's fan draws about 30mA, and measurements can be collected basically as soon as air enters the unit's chamber. Therefore, in a battery-powered unit that only measures at hourly or daily time intervals, this sensor doesn't consume much power from the battery.

CO_2 SENSOR

There are several different types of CO_2 sensors available at reasonable prices. The first type that I ran into were labeled as VOC sensors, and also measured CO_2 eq (equivalent) as a "side-hustle," so to speak. VOC stands for volatile organic compound. Sensirion made such a unit: the SGP30. They are now obsolete and their replacement, the SGP40 now only claims to measure VOC. From that I assume that the SGP30's CO_2 eq readings were not particularly accurate or useful. The SGP40 is a MOx based gas sensor meant for indoor air quality testing. Sensirion doesn't go into details about how they work but suffice it to say that this sensor requires that a heater come on to heat up this MOx CMOS substrate, and it takes some time for the sensor to register properly.

There are now better ways to measure CO_2 using reasonably priced small sensor modules. There are two different devices that I am aware of:

- The Sensirion SCD40 (400-2000 ppm) and the SCD41 (400-5000 ppm)
- The Infineon Xensiv PAS-CO2

Both units measure the actual CO_2 concentration and do not just provide an educated guess like the SGP30 provides. Also, they don't require turning on a heater

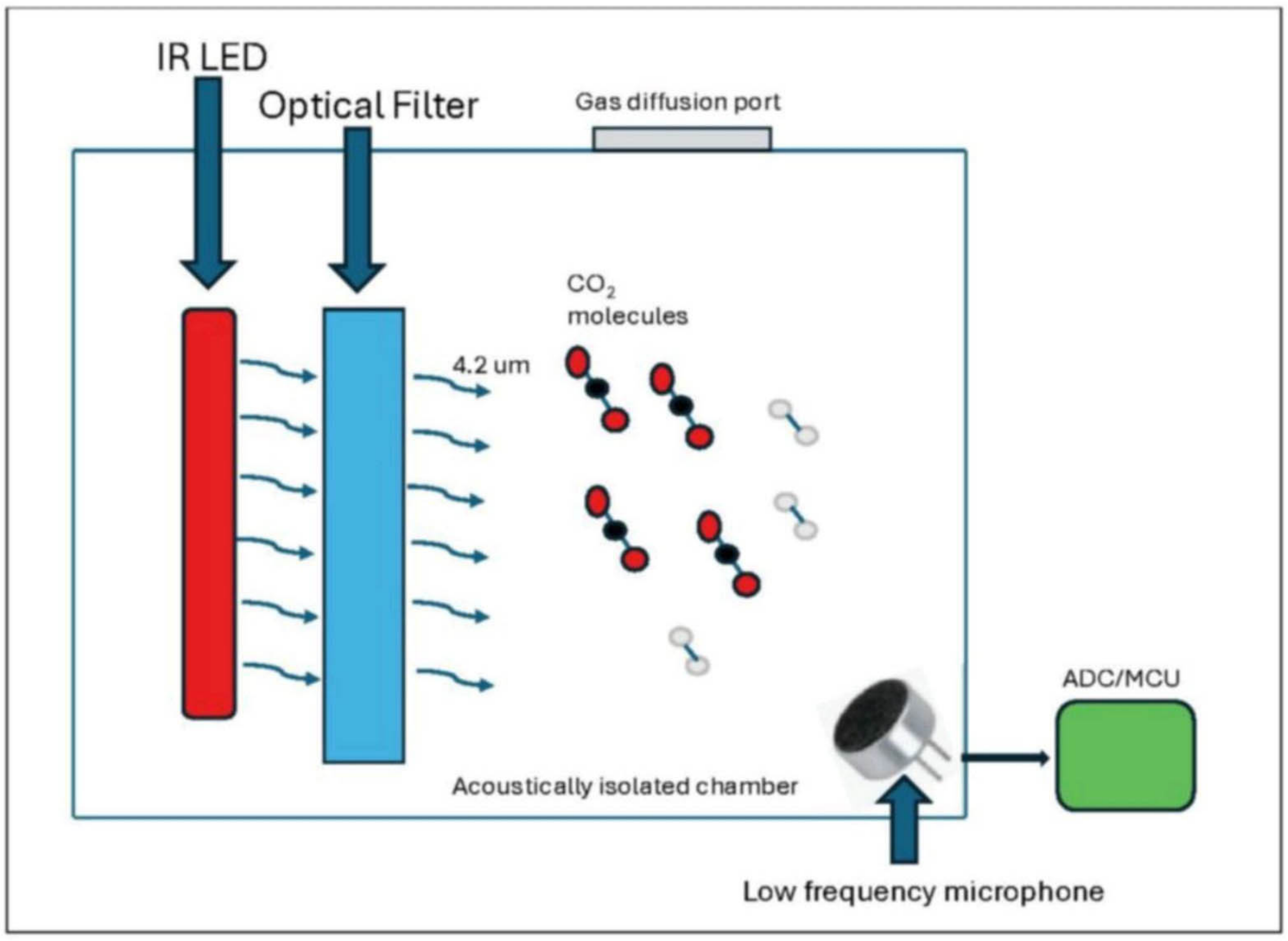


FIGURE 5
This is a block diagram showing how the Infineon Xensiv PAS-CO2 CO_2 monitor works. This is a very sophisticated method of measuring CO_2 compared to earlier Air Quality sensors made by Sensirion and others and is much more accurate.

and waiting until a CMOS substrate is up to temperature. They basically provide a reading within a second or so.

Both sensors use the photoacoustic spectrometry (PAS) method. This method was discovered more than 100 years ago by Alexander Graham Bell. Please refer to **Figure 5**, for a visual explanation about how this works:

Inside the sensor is a chamber that has an opening to the outside air but contains a dust filter to only allow gases to penetrate. An Infrared LED is activated in a series of very short pulses, and this light is further filtered to allow only a narrow band of IR light to pass through. The wavelength of this narrow IR light beam is selected to be one that will be absorbed by a specific gas- in the case of CO_2, λ=4.2µm. This causes those gas molecules to heat up and cool down quickly, which causes small pressure changes in the chamber. This can be picked up by a tiny, low-frequency microphone and that signal can be processed and converted into CO_2 concentration. The accuracy of this reading is affected by air pressure, but this is compensated for internally in the sensor itself.

Of the two sensors I've mentioned, the size and price of the Infineon Xensiv PAS-CO2 were advantageous to me, so I chose that model. **Figure 6** shows this module. While it is not clear from Figure 5, this module needs two power supplies to operate:

- A 5V supply capable of supplying around 265mA, in a short burst, for the IR LED
- A 3.3V supply capable of supplying 10mA, which powers the rest of the sensor circuitry

I'll discuss the power supply that I used later in the article. However, I'll mention that the 3.3V power required for this sensor, uses a low enough current that I supplied it via one of the ESP32C3's GPIO pins. That means that this sensor is basically powered down most of the time and only comes on when an actual measurement is taken. The 5V for the IR LED is controlled by a MOSFET inside the sensor itself and is only activated for a short burst during the measurement itself.

The Infineon PAS-CO2 sensor contains both an I^2C and a UART interface. In this project I am using the ESP32C3's UART port for the PM1006K PM2.5 sensor. Therefore, I used the PAS-CO2's I^2C port to connect to the ESP32C3.

Infineon provides a programming guide for the PAS-CO2, which includes all the low-level commands, register addresses, and such needed if you want to write your own driver code. However, they also provide libraries for both Arduino, and their own MCU- the Cypress (Now Infineon) PSoC6 MCU (found on the WiFi-BT Pioneer kit). I used the Arduino library and even though it was written for an Arduino Due containing an Atmel SAM3X8E ARM Cortex-M3 MCU, it worked fine with the ESP32C3.

CONTROLLER CIRCUITRY

I've already discussed the operation of the PM2.5 and CO_2 sensors used in this project. While not exactly related to air quality, I had a Silicon Labs SI7021 Temperature and Humidity sensor on hand, so I added it to the circuit. It also uses the I^2C bus and therefore didn't occupy any more GPIO lines on the ESP32C3 than I had already utilized. This sensor is much more accurate, especially in terms of humidity, than many of the other humidity sensors that are available at low prices.

Figure 7 shows a schematic diagram of the controller. This contains all the circuitry apart from the Blues Starnote and WiFi Notecard- both of which are mounted on the Carrier-XS base board. The only connection between this controller board and the Carrier XS are:

- I^2C connection
- VMAIN, which is a source of 3.6V for the controller, from the LiPo battery plugged into the Carrier XS. The Carrier XS board contains a BQ25628 LiPo charger chip, which contains a low RDS MOSFET switch

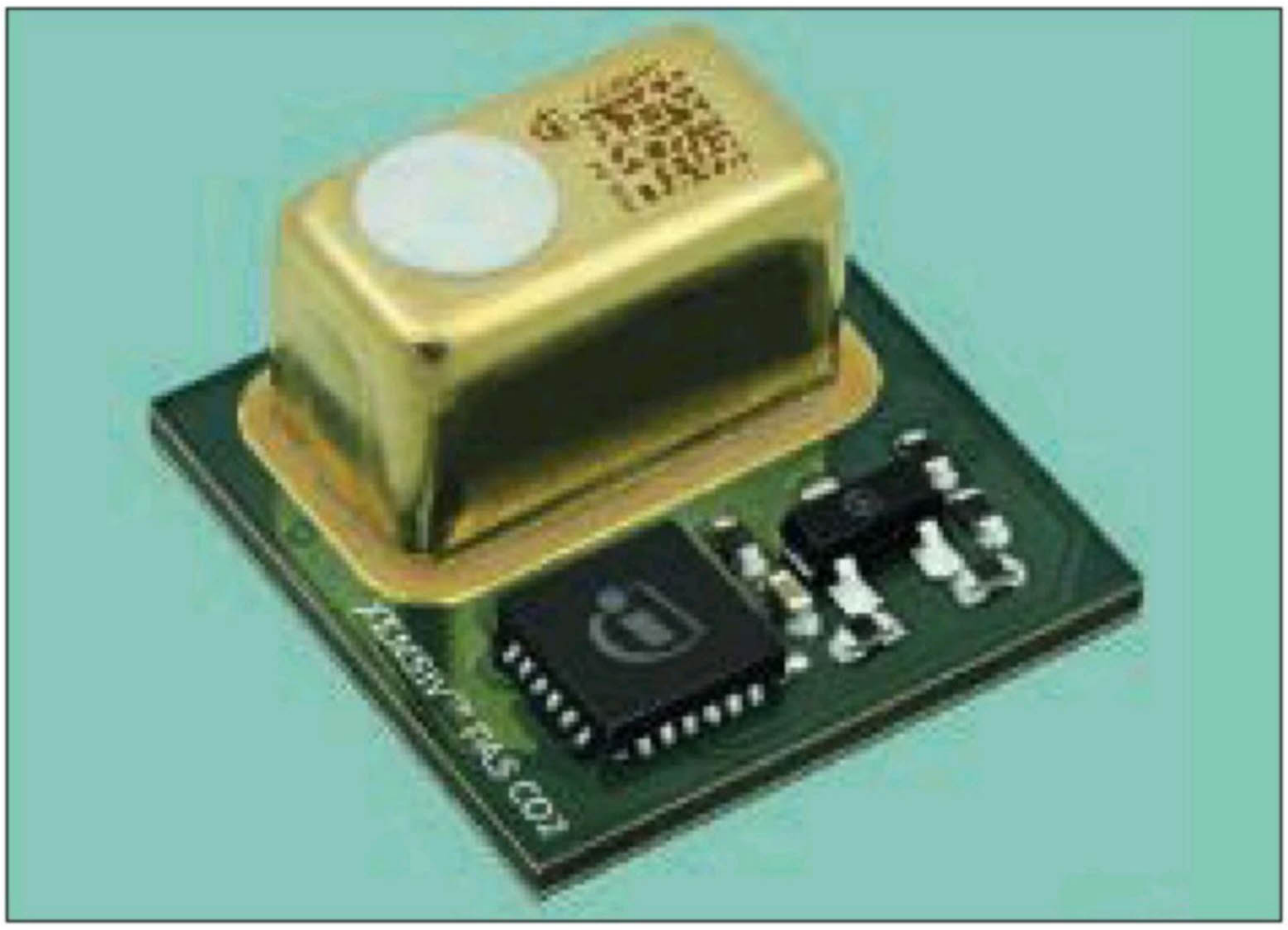
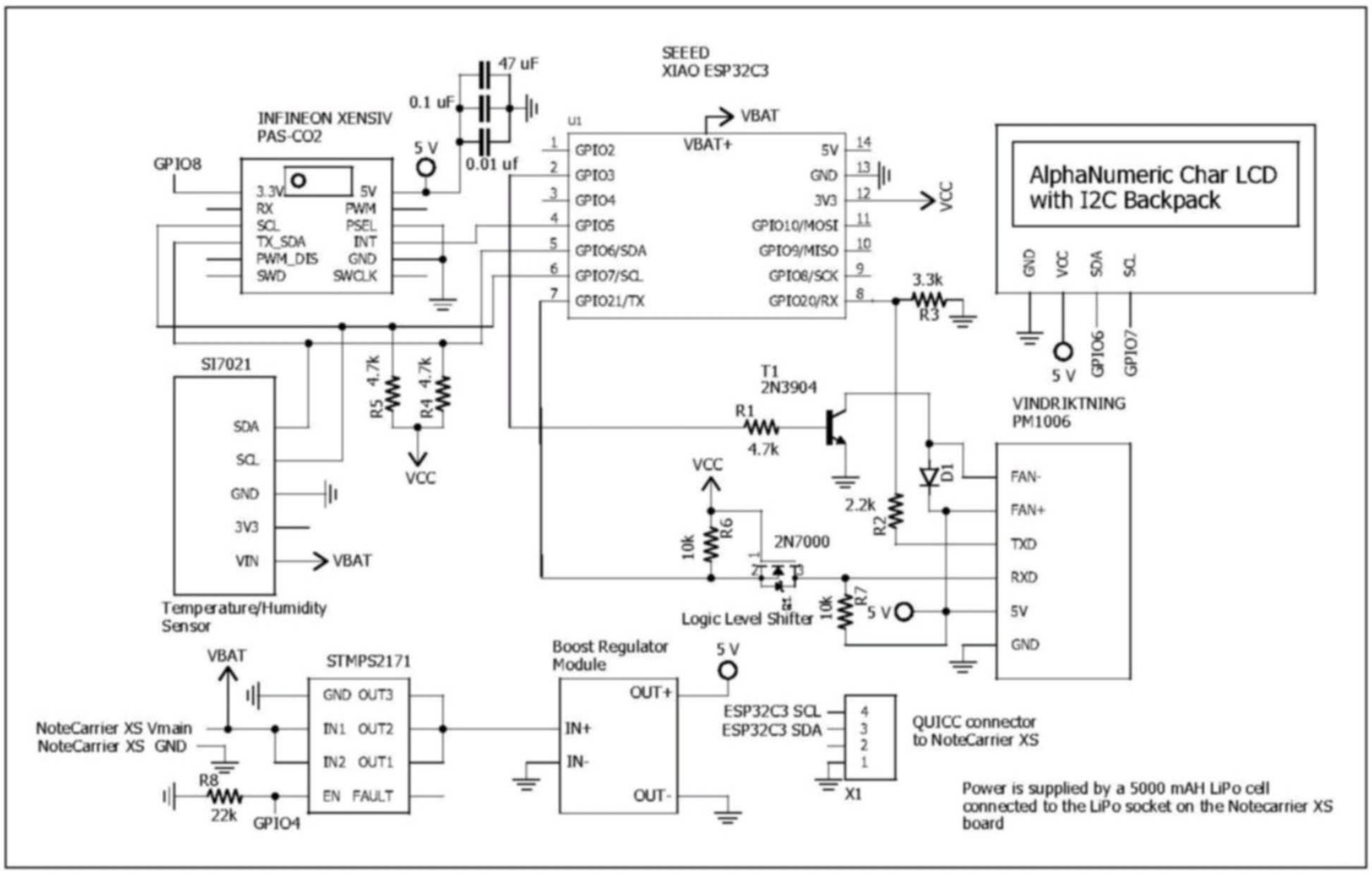


FIGURE 6
THIS IS A PHOTO OF THE INFINEON PAS-CO2 SENSOR. THE PCB IS ABOUT 1 INCH SQUARE.

ABOUT THE AUTHOR

Brian Millier runs Computer Interface Consultants. He was an instrumentation engineer in the Department of Chemistry at Dalhousie University (Halifax, NS, Canada) for 30 years.

FIGURE 7
This is a schematic diagram of the project. Everything is powered by a 5000mAh Lipo cell plugged into the Notecarrier XS board. A boost converter provides 5V needed by some of the circuitry, and is switched on as needed.


SEEED
XIAO ESP32C3
U1
VBAT
VBAT+
47 uF
0.1 uF
0.01 uf
INFINEON XENSIV
PAS-CO2
5 V
GPIO8
3.3V
RX
SCL
TX_SDA
PWM_DIS
SWD
5V
PWM
PSEL
INT
GND
SWCLK
1 GPIO2
2 GPIO3
3 GPIO4
4 GPIO5
5 GPIO6/SDA
6 GPIO7/SCL
7 GPIO21/TX
5V 14
GND 13
3V3 12
GPIO10/MOSI 11
GPIO9/MISO 10
GPIO8/SCK 9
GPIO20/RX 8
VCC
3.3k
R3
AlphaNumeric Char LCD
with I2C Backpack
GND
VCC
SDA
SCL
5 V
GPIO6
GPIO7
SI7021
SDA
SCL
GND
3V3
VIN
VBAT
Temperature/Humidity
Sensor
R5 4.7k
R4 4.7k
VCC
T1
2N3904
R1
4.7k
VCC
R6
10k
2N7000
Logic Level Shifter
R7
10k
5 V
2.2k
R2
D1
VINDRIKTNING
PM1006
FAN-
FAN+
TXD
RXD
5V
GND
VBAT
STMPS2171
GND OUT3
IN1 OUT2
IN2 OUT1
EN FAULT
NoteCarrier XS Vmain
NoteCarrier XS GND
R8
22k
GPIO4
Boost Regulator
Module
5 V
OUT+
IN+
IN-
OUT-
ESP32C3 SCL
ESP32C3 SDA
4
3
2
1
X1
QUICC connector
to NoteCarrier XS
Power is supplied by a 5000 mAH LiPo cell connected to the LiPo socket on the Notecarrier XS board

that will disconnect the LiPo battery if its voltage drops too low (a LiPo battery can be rendered useless if it is allowed to discharge too much).

While the controller board is supplied with only 3.6V from the Carrier XS board, it requires 5V for the PM1006K PM2.5 module, the IR LED on the PAS-CO2 module and the SI7021 sensor. While not entirely necessary, I added a 4-line × 20-char LCD alphanumeric display containing an I^{2}C backpack, which also needs 5V. This was useful during development, especially as a debugging tool when sending data to Notehub using the Starnote satellite board. As mentioned before, this requires the Starnote antenna to be outdoors, or in my situation, near a south-facing picture window. In neither case do I have a computer handy to monitor any debugging messages sent through the USB-serial port on the ESP32C3.

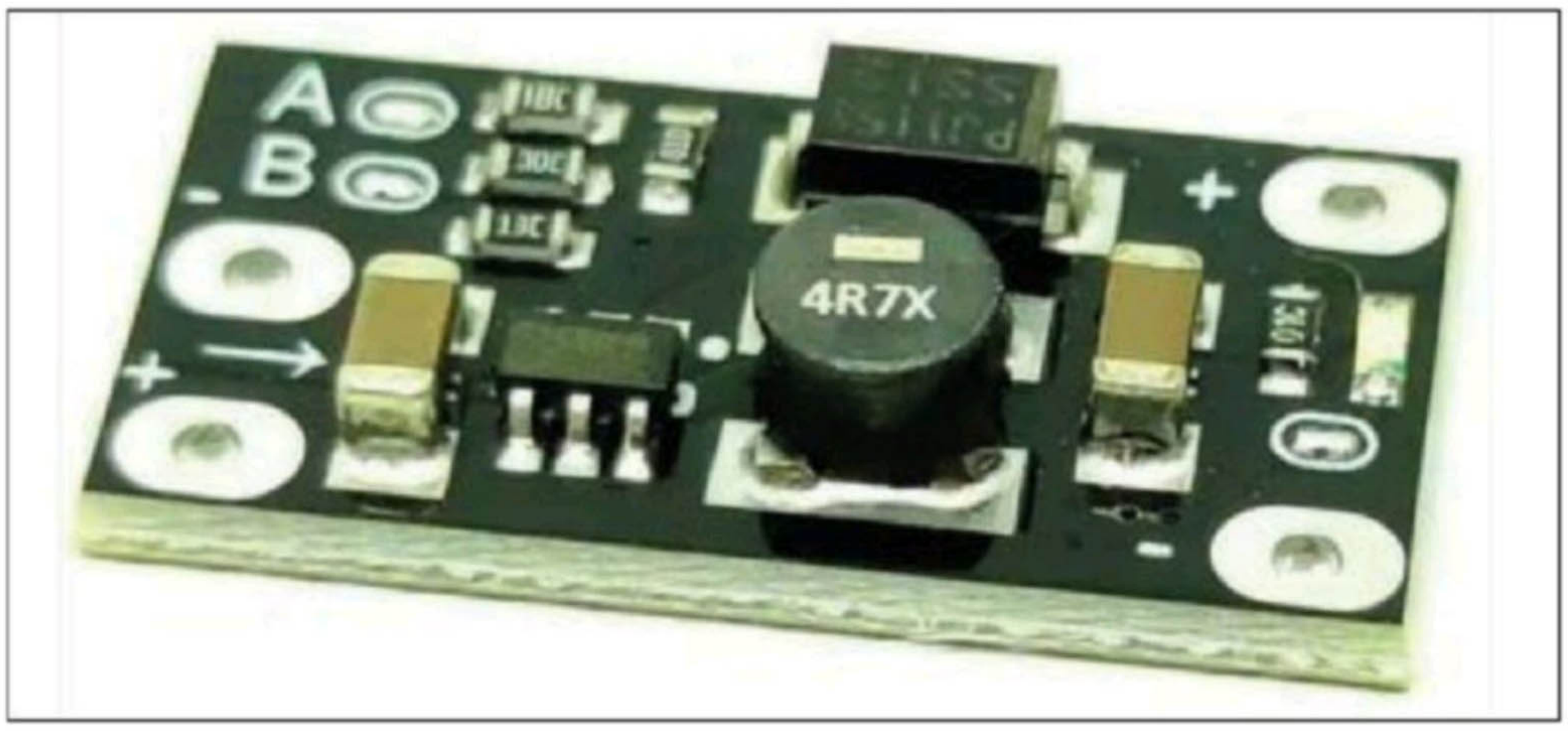


Figure 8
This is a generic Boost regulator that provides 5V needed by some circuitry. It can be obtained from Amazon.

To get 5V from the 3.6V available from the LiPo battery, I needed two things:

- A boost converter
- A high-side power switch to turn on the boost regulator only when needed (i.e., during sensor measurements)

I used a small boost converter as shown in **Figure 8**. This converter takes an input voltage of 2.5-5V and converts it into 5-8-9-12V, jumper selectable. It can handle 6W, much more than needed here. I bought mine from a Canadian source, so that part number is not meaningful to readers, but they are commonly available online.

The boost converter will draw some power even with no load. Since this would account for most of the time, it made sense to remove the input voltage except when 5V was needed during measurements. I used an STMicroelectronics STMPS2171 high-side power switch to do this. This device draws a negligible amount of current when it is not turned on. The active high Enable line is connected to the ESP32C3's GPIO4 line and is pulled low by R8 whenever the ESP32C3 is in Sleep mode.

Level shifting of the ESP32C3's 3.3V logic to the PM1006K's 5V logic is performed by R2/R3 and the circuitry associated with Q1, a 2N7000 MOSFET.

The I^{2}C signals from the controller to the Carrier XS board are handled by a 3-conductor cable going to the Carrier's QUICC socket. Note that the Vcc pin is not connected in this case, as the power for the controller comes from the V_{main} pin on the Carrier XS board.

Figure 9 shows the controller board, LCD display, the PM1006K sensor, and the Carrier XS board with both the Starnote and the WiFi Notecard mounted. Everything is mounted in a waterproof enclosure meant for electrical use and obtained from Amazon. Not shown is the satellite "patch" antenna, which is mounted on the other (larger) half of this enclosure in such a way that it is oriented properly to reach the Skylo satellite.

POWER CONSUMPTION

I made some design decisions based upon the desire to have the unit able to operate on a LiPo battery for a reasonable amount of time. My first assumption was that I would like to take readings every 4 hours since air quality would not generally change too much over that period of time. Since I could be sending the readings to the Skylo satellite using the Starnote card, there was the additional constraint that the Starnote card came with a satellite traffic allocation of 18,000 bytes. Even though the Blues "notes" that transmit the sensor data are compact, the 18,000-byte limit would be used up quickly if I were to take readings at sub-hourly intervals.

This project contains three separate MCUs:

- A ESP32C3, on a Xiao board, acting as the main controller
- An STM32L4R5ZI ultra-low-power ARM Cortex M4 on the Notecard WiFi
- An undisclosed MCU running on the Starnote card

If you wanted to be more precise, there is an unspecified MCU contained in the PM1006K PM2.5 particle sensor as well as an MCU, likely by Infineon, contained on the Infineon PAS-CO2 sensor module.

Please refer to **Figure 10** for the screen capture of the Joulescope power measurements that I performed. The project was designed to draw low battery power using the following design concepts:

- The ESP32C3 is placed in a deep-sleep mode for the 4-hour interval between measurements. In deep-sleep mode the ESP32C3's internal RTC is still operating, (as well as some low-power SRAM associated with the RTC). It is this RTC that wakes up the ESP32C3 at 4-hour intervals to perform the measurements. I chose the Xiao ESP32C3 module as it contains virtually no circuitry that is powered by the LiPo battery input—apart from the ESP32C3 itself, which draws <20µA in deep-sleep. Any other circuitry on this module is powered by the USB-C port, but only when that is connected.
- The STM32L4R5ZI ARM Cortex M4 on the WiFi Notecard normally runs in an idle state. Blues claims that it consumes <8µA in this state and yet can respond to activity on its I²C port, which is how commands are sent to it from the ESP32C3.
- The Starnote card also runs in an idle state normally. Blues claims it consumes <4µA in this state. Since the Starnote is strictly controlled by the WiFi notecard, it only gets activated when the WiFi notecard has a "note" to send to the Notehub via satellite. For this project I'm not looking for any "notes" coming back from the NoteHub, so the WiFi notecard is not waking up periodically to check for these.
- The Carrier XS board, which houses both the WiFi Notecard and the Starnote card, contains very little circuitry that draws any appreciable amount of power. It does have a BQ25628 Battery Charger containing power path circuitry, which disconnects the LiPo battery from the load if the LiPo voltage falls below a safe threshold. This device itself draws only a few microamps (µA) when in the battery only mode.
- The PM2.5 and CO_2 sensors, as well as the LCD display, require a 5V supply. This 5V supply is provided by a DC/DC boost regulator, fed by the LiPo battery (through the BQ25622 power path MOSFET). A STMPS2171 high-side switch only connects the boost regulator to this power source for ~ 8 seconds that the actual measurements are being taken. The

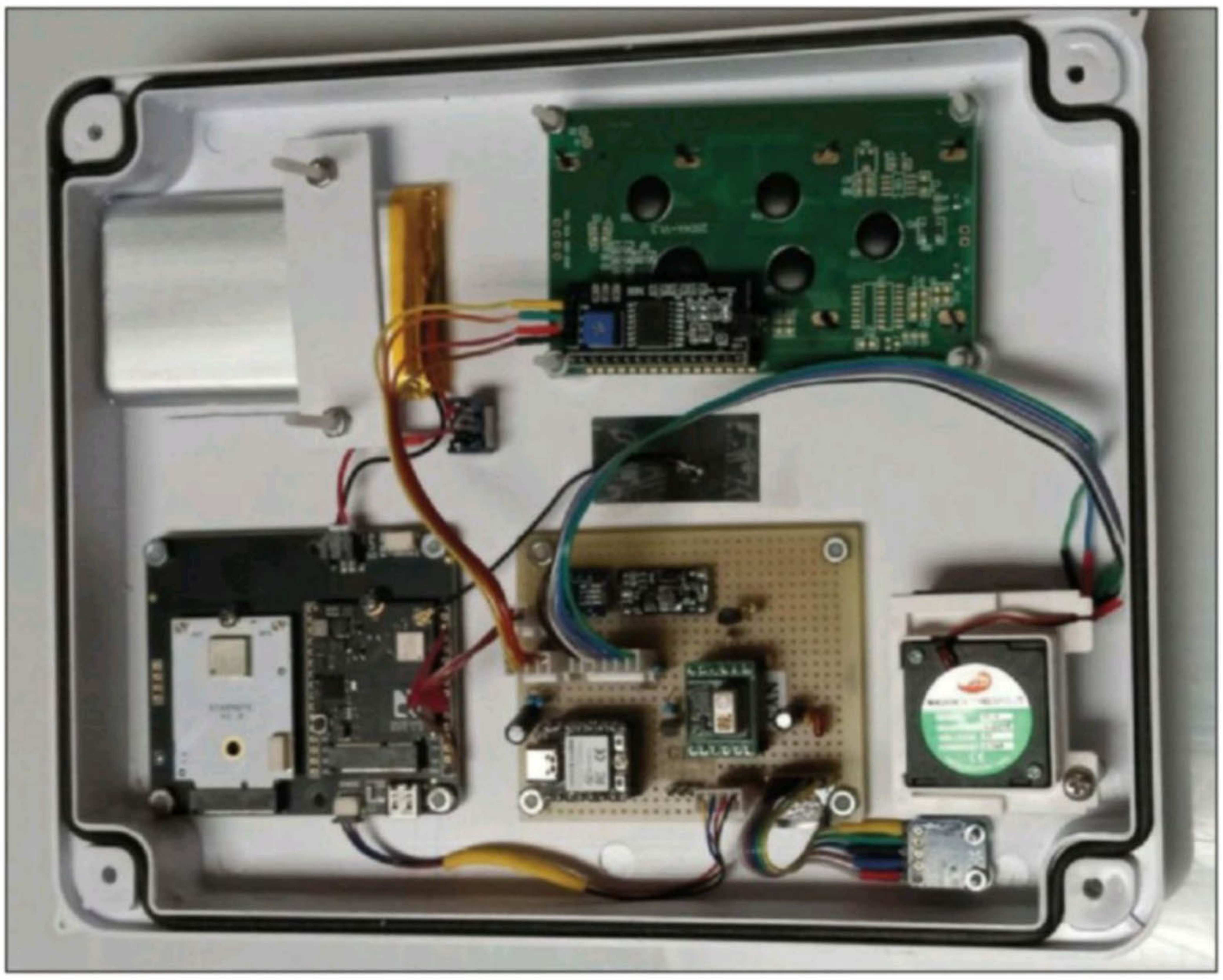


FIGURE 9

This is an interior view of the project, which is mounted in a waterproof plastic cabinet. The Quectel antenna is not visible as it mounts on the other section of the enclosure and is oriented to "see" the Skylo satellite constellation.

STMPS2171 itself takes negligible power when it is not enabled.

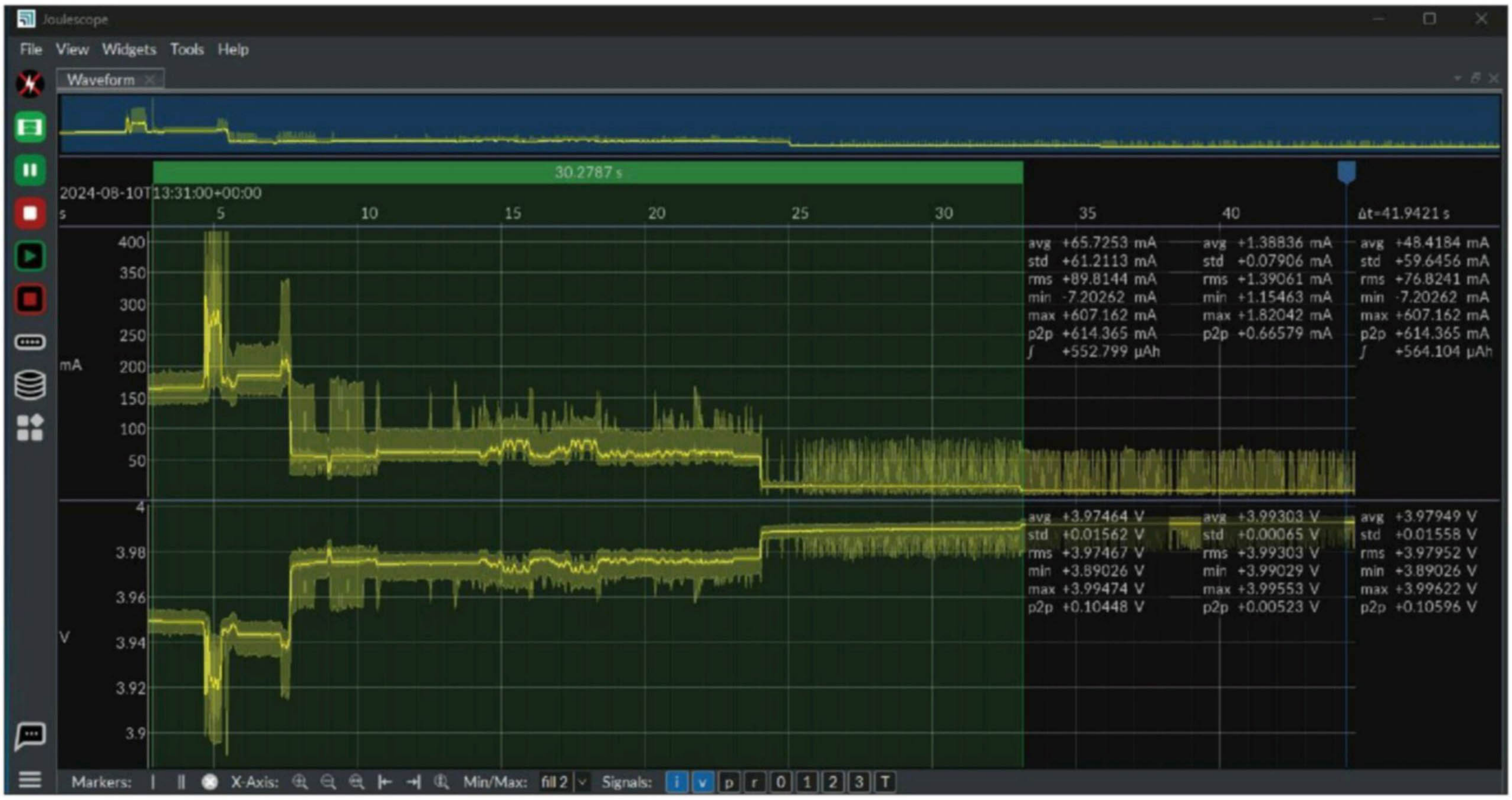
- The SI7021 temperature and humidity sensor draws only 150µA in operation, and 60nA in standby. When the ESP32C3 wakes up, it will check this sensor first to see if the humidity is so high that I don't want to subject either the PM1006K or the PAS-CO2 sensors to this moist air. In this case, I don't activate the PM1006K's fan, so I'm not drawing in any outside air. No CO_2 nor PM2.5 readings are taken in this case. For this scheme to work, the SI7021 must be powered by something other than the switched 5V power supply—it's instead fed from the Carrier XS board's VMAIN pin (which is connected to the VBAT bus in the schematic). This is the LiPo voltage gated by the internal MOSFET in the BQ25628.

Although the sum of all the idle currents mentioned here is still in the hundred µA range, there are some other passives (e.g., pull-up resistors, etc.), which increase it somewhat. Looking at Figure 10, the readout from my Joulescope power analyzer, I dropped the blue vertical cursor in the idle (non-measurement) interval. The average current is 1.38mA. This will be drawn for all but about 30 seconds of the 4-hour measurement interval.

The first half of the display shows the current draw during the 30.3 second measurement period. For the first 7 seconds, the current averages 170mA with short 600mA spikes as well as 330mA spikes at about 7 seconds. Half of this current can be attributed to the ESP32C3's normal run current. The other half can be attributed to the PM1006K and PAS-CO2 sensors as well as losses in the boost regulator. During this interval both the ESP32C3 and the WiFi Notecard are running, and the two groups of current spikes are due to the PAS-CO2 IR LED pulses and RF transmissions by the WiFi Notecard.

For this measurement, I am not using Starnote to transmit to the satellite, just WiFi. To contact the satellite, the project must be outdoors (or near a south-facing picture window), and at those locations, I can't use the Joulescope and its host PC to measure the power consumption. All I can assume is that the Starnote's Quectel L76-L modem is not using a whole lot more power than the WiFi Notecard's WiFi Modem. Blues doesn't give power consumption figures for the Starnote module.

After about 8 seconds, the ESP32C3 goes into a deep-sleep mode and the average current consumption drops from ~170mA to ~55mA. Beyond this point, the Notecard WiFi is sending the message out, and at around 23 seconds it drops to around 5mA. At about the 30 second mark, the Notecard WiFi goes into idle mode and the current now drops to 1.39mA—for the remainder of the 4-hour period. I can't break out the current consumption of the various modules, with the



```
avg  +65.7253 mA    avg  +1.38836 mA    avg  +48.4184 mA
std  +61.2113 mA    std  +0.07906 mA    std  +59.6456 mA
rms  +89.8144 mA    rms  +1.39061 mA    rms  +76.8241 mA
min  -7.20262 mA    min  +1.15463 mA    min  -7.20262 mA
max  +607.162 mA    max  +1.82042 mA    max  +607.162 mA
p2p  +614.365 mA    p2p  +0.66579 mA    p2p  +614.365 mA
∫    +552.799 µAh                       ∫    +564.104 µAh

avg  +3.97464 V     avg  +3.99303 V     avg  +3.97949 V
std  +0.01562 V     std  +0.00065 V     std  +0.01558 V
rms  +3.97467 V     rms  +3.99303 V     rms  +3.97952 V
min  +3.89026 V     min  +3.99029 V     min  +3.89026 V
max  +3.99474 V     max  +3.99553 V     max  +3.99622 V
p2p  +0.10448 V     p2p  +0.00523 V     p2p  +0.10596 V
```

FIGURE 10

This is a screenshot of the Joulescope Power Analyser's PC computer application, during a measurement/upload interval.

Joulescope, but I am making these assumptions based upon the status LEDs on the Notecard WiFi and Starnote modules.

The total power consumption for 1 day (using the figures obtained from the Joulescope in Figure 10), would be:

Measurement process: 553µAh x 6
measurements/day = 3318µAh = 3.318mAh
Deep-sleep intervals: 1.388mA x 24 hours = 33.312mAh
Total = 36.7mAh

Using a 5000mAh LiPo battery, the unit should be able to run for 136 days without depending upon power from a 5V power source plugged into an exterior receptacle of the house.

In measuring the power consumption, I became aware of something that could affect power consumption that I wasn't able to take into consideration. The Starnote Starter kit that I received, comes with a WiFi V1.2 Notecard (in addition to the Starnote and Carrier XS boards). As mentioned earlier, this WiFi Notecard contains the very low-power STM32L4R5ZI ARM MCU, which can run in an idle mode from which it can still respond to commands from the ESP32C3 controller in this project (via I^2C). While this WiFi Notecard is supplied with the Starnote Starter kit, it is no longer available, on its own, from Blues. Instead, the currently available WiFi module is the NOTE-ESP WiFi module, which uses an ESP32S3 MCU. The following is what Blues says about the differences in low-power mode between the two modules.

Unlike other STM32-based Notecards, NOTE-ESP does not (by default) fall back to a ~10µA current draw when it is idle. This is because the ESP32 doesn't have a STOP mode equivalent where the UART and I^2C can operate. The ESP32 does have a deep sleep mode that draws ~10µA at 5V, but in this state:

- The RTC is retained.
- GPIOs are retained.
- It can wake based on RTC or GPIO.
- UART and I^2C are not functional.

To enable the NOTE-ESP to enter a deep sleep state during periods of inactivity, the host must configure it with the card, sleep API, and drive RTX high to wake the NOTE-ESP from deep sleep when it wants to communicate via UART or I^2C. Note that there is significant latency on wake (~10 seconds). The NOTE-ESP sets CTX high whenever it is ready to receive commands via I^2C or UART.

As I didn't have this NOTE-ESP board, I didn't allow for the considerations mentioned in the preceding paragraph.

THE ESP32C3 CONTROLLER FIRMWARE

As with all Notecards, this project needs an MCU controller to both handle the sensors and more, and to pass on this information to the Notecard to be sent out as a "note" to Notehub. Please refer to **Figure 11** to see what is going on in the ESP32C3's firmware. The SI7021 is measured first and if the humidity is >85%, then it's better not to bring in such moist air to the project's enclosure. Therefore, the small fan in the PKM1006K is not activated and dummy values of 9999 are assigned to CO_2 and PM2.5. I use dummy values of 9999 since I found that the Notehub dashboard does not show a variable's numeric value if it is zero. This is the default behaviour. Although I wasn't aware of it at the time, you can over-ride this, as described in the Notecard Walkthrough documentation, in the Low-Bandwidth Design/ Use of Templates section. Next, the humidity, temperature, and LiPo battery voltage readings are sent to the Notecard along with the dummy values for CO_2 and PM2.5.

If the humidity is <85%, the PM1006K's fan is turned on and after 5 seconds, the PAS-CO2 sensor is read. After that, the PM2.5 value is read out from the PM1006K. Those two values, plus temperature, humidity, and the LiPo are sent to the Notecard. The LiPo voltage is measured by the Notecard, using the notecard.newRequest("card.voltage") API. The Notecard will send this note to Notehub, via WiFi or Cellular, depending on what Notecard board is mounted. If that transmission is unsuccessful, the Notecard will perform a "fail-back" routine, which sends the same note to the Skylo satellite using the Starnote card.

Once the five values are sent to the Notecard, it will handle the transmission of the note independently, using the Starnote/Skylo satellite, if necessary. Therefore, after passing the note on to the Notecard, the ESP32C3 can be placed in a deep-sleep mode. The ESP32C3's wakeup routine uses its internal RTC, which is set for a 4-hour interval in the program, using the following directive:

```
#define TIME_TO_SLEEP  14400   // 4 hours
```

As I mentioned early in the article, before these "notes"

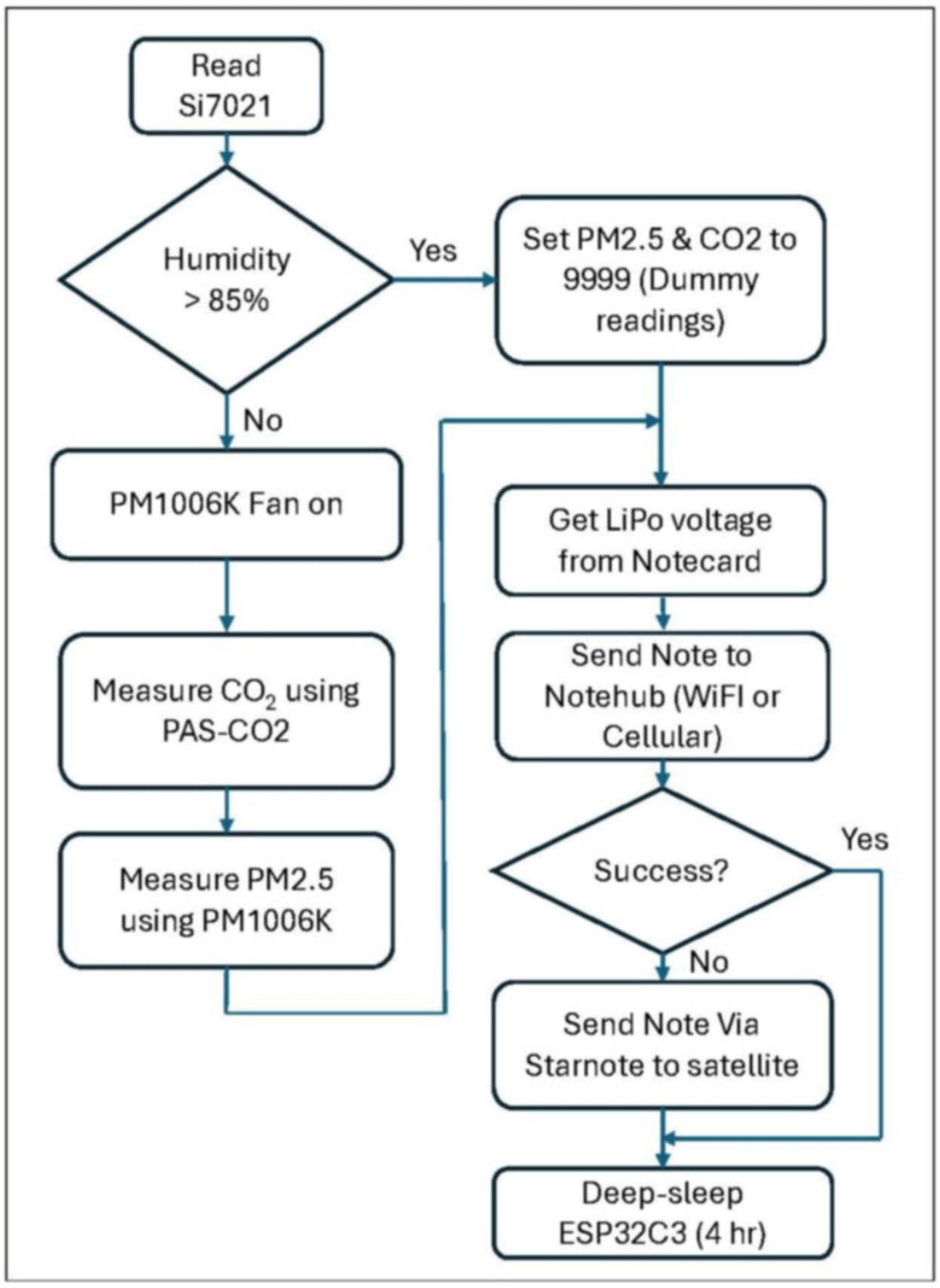

Read Si7021
Humidity > 85%
Yes
Set PM2.5 & CO2 to 9999 (Dummy readings)
No
PM1006K Fan on
Measure CO₂ using PAS-CO2
Measure PM2.5 using PM1006K
Get LiPo voltage from Notecard
Send Note to Notehub (WiFI or Cellular)
Success?
Yes
No
Send Note Via Starnote to satellite
Deep-sleep ESP32C3 (4 hr)

FIGURE 11
This is a flowchart of the operation of the environmental monitor.

can be sent to Notehub, there must be a template established for this data, and the notecard must be “synced” with the Notehub. Initially, I did this by manually typing in the necessary JSON commands in the “in-browser terminal” provided by Blues—with the Notecard connected to my PC via the USB cable.

However, I also included the code to perform this template definition in the project’s firmware as follows:

```
if (FIRST_RUN) {
 InitialHub_Set();
 }
```

Unless the user has manually defined the template and synced the Notecard to Notehub, the constant “FIRST_RUN” must be set to true for the first time that the program is run. Also, one would want to have the Notehub dashboard running on your PC, to ensure that this initial message was received by Notehub—thus “Syncing” the Notecard to the Notehub. After that, FIRST_RUN should be changed to false, in the code. Otherwise, the more-involved template definition/sync procedure would be repeated every 4 hours, which is unnecessary.

Since the Starnote board is only used if the Notecard is unable to transmit its notes to the Notehub, it must be tested some other way—assuming that your WiFi connection is good, or cellular reception is adequate, depending upon the Notecard mounted.

When a Notecard and the Starnote are both mounted on the Carrier XS board, the two modules are interconnected by a UART link (labelled NC Rx, NC Tx in the schematic). The Notecard is aware of the presence of the Starnote via this UART link. So long as the Notecard can connect to the Notehub via its native mode (WiFi or Cellular), it won’t use Starnote for such communication. However, if it fails to connect to the Notehub, after a preset number of retries, it will then communicate with the Starnote board and send the message that way. This procedure is called “fail-back.” I noticed that the block diagram that Blues supplies along with the rest of the schematic package for the Notecarrier-XS, shows an I²C link between the Notecard and Starnote modules. It is actually a UART link, which I reported, and Blues is updating the schematic document.

To test the ability of Starnote to connect to the Skylo satellite constellation, you don’t need to make any substantial changes to your host MCU program. All that is necessary is to send the flowing command to the Notecard:

```
{“req”:”card.transport”,”method”:”ntn”}
```

After receiving this command, the Notecard will send all its notes via the Starnote module/Skylo satellite.

This command could be sent using your host MCU’s program or more easily by using the Blues In-Browser Terminal, with the Carrier-XS board plugged into your PC via the USB port. Note that this is a non-volatile command—this mode will persist after a power-down cycle. So, you would have to issue something like the following to revert to normal “fail-back” operation (tailored to whatever Notecard you have):

```
{“req”:”card.transport”,”method”:”wifi-ntn”}
```

This procedure is well-described in the Satellite Quick Start guide, in the “Understanding NTN Transport Mode” section. This can be found on the Starnote Starter Kit’s webpage on the Blues online store.

CONCLUSIONS

Initially, I questioned the effectiveness of having to use a Starnote, a Notecard and a host MCU controller to implement an IoT device with WiFi/Cellular connectivity and satellite failback. One concern was whether this would require more power than other solutions, which would be an issue in battery-operated devices. Also, I wondered about cost.

However, I believe that the upside to doing it this way is significant. I’m experienced with the ESP32 MCUs, which contain an internal WiFi modem and an RF protocol stack that is basically insulated from the user’s program since it’s burnt into Rom and executed on its own processor core. However, in many cases, engineers will have a lot of experience with specific MCUs that are not RF-enabled and want to write their IoT controller code in a language/IDE with which they are familiar. Also, the transportation mode that they need may have to include WiFi, Cellular, or both, as well as the satellite failback provision.

In these cases, the Blues Notecard/Starnote modules simplify the process measurably. Once the host controller has processed any sensor data, it merely embeds the applicable data into a JSON packet and sends it to the Notecard to handle. The host controller also needn’t stay in an active mode continuously, if it is expecting randomly timed incoming data (for actuators, etc.). The Notecard/Starnote will take care of this and can bring the host controller out of sleep mode when such data arrives. As I discussed in the Power Consumption section, the power requirement of the Notecard/Starnote module combination is very low and the Notecard is capable of waking the host MCU from sleep mode, so the total consumption levels are well within what is needed for a battery-powered application.

Along with the other considerations, I found the Blues documentation to be excellent. In particular, the integration of the In-Browser Serial Terminal into the Quickstart guides, makes it easy to try out the various concepts on the hardware itself, as each of these concepts are introduced in the guide.

Author's Note: I’d like to acknowledge the assistance supplied by Tre Clay, Rob Lauer, and Desiree Caudillo from Blues, in preparing this article.

Additional materials from the author are available at:
www.circuitcellar.com/article-materials

Resources

Arduino | www.arduino.cc

Blues | https://dev.blues.io

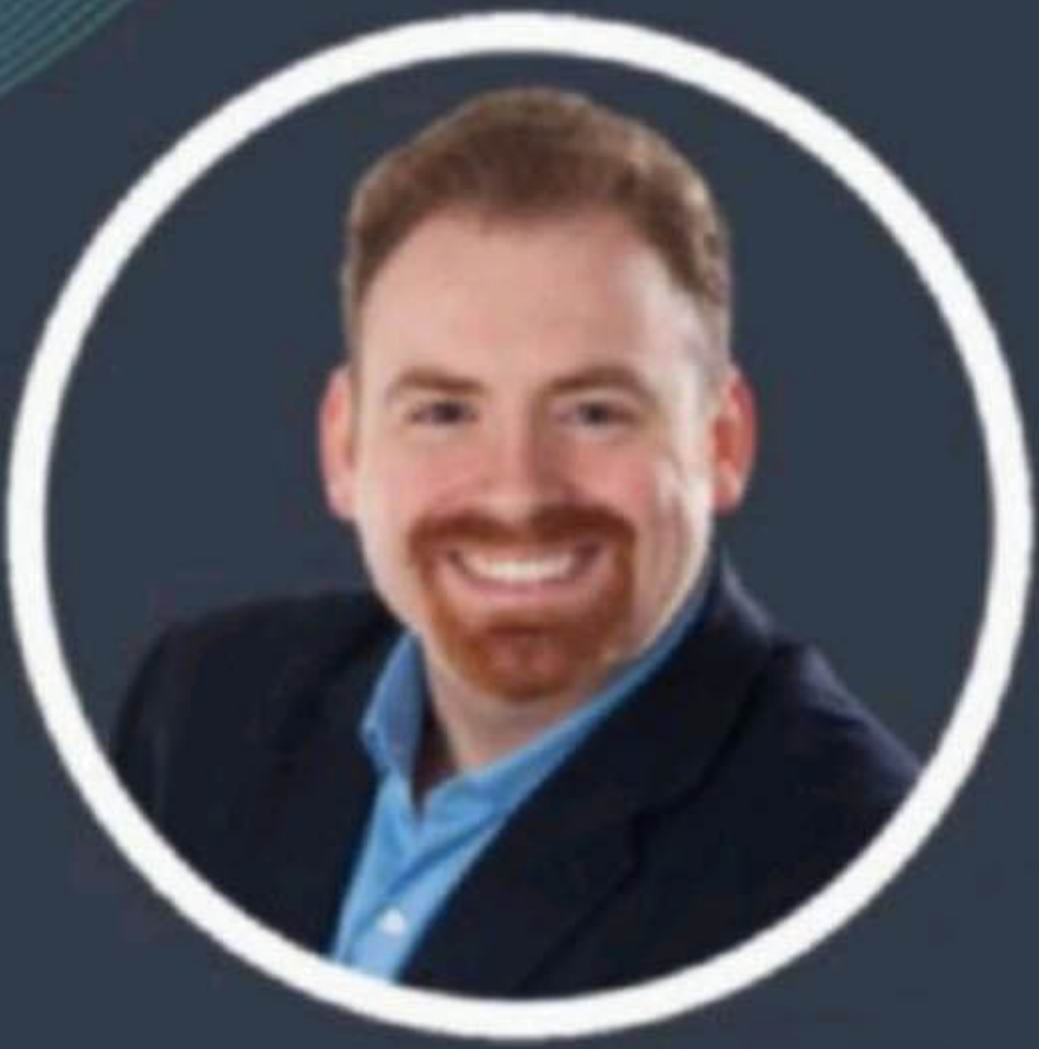
Infineon | www.infineon.com

Quectel | www.quectel.com

Silicon Labs | www.silabs.com

STMicroleectronics | www.st.com

Skylo Technology | www.skylo.tech



JACOB BENINGO
CEO & Founder

EMBEDDED SOFTWARE ACADEMY BY BENINGO

Embedded
Software Academy

Get Expert Guidance to Start, Grow and Modernize your Embedded Software Skills

Save Time **Save Money** **Advance Career**

Learn More: → **www.beningo.com**

The Premier Online Conference for Embedded Systems Engineers

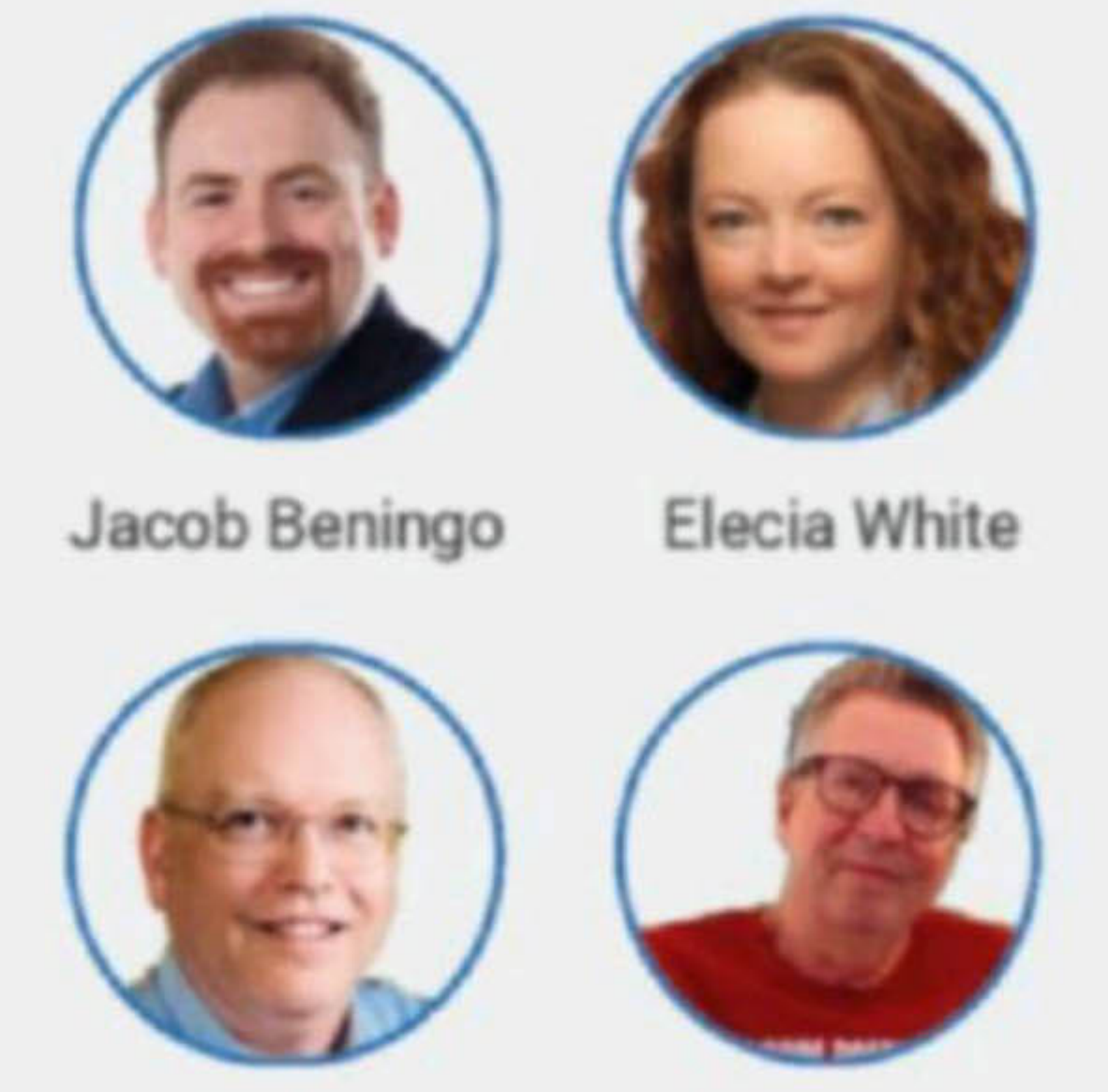
Advance Your Skills. Accelerate Your Career.

- ✓ Expert-led training at an unbeatable value
- ✓ Learn on your schedule—unlimited on-demand access until Dec 2025
- ✓ Actionable insights you can apply immediately

Don't fall behind—stay ahead with the latest in embedded systems.

Embedded Online Conference

KEYNOTE SPEAKERS



Jacob Beningo
Elecia White
Philip Koopman
James Grenning

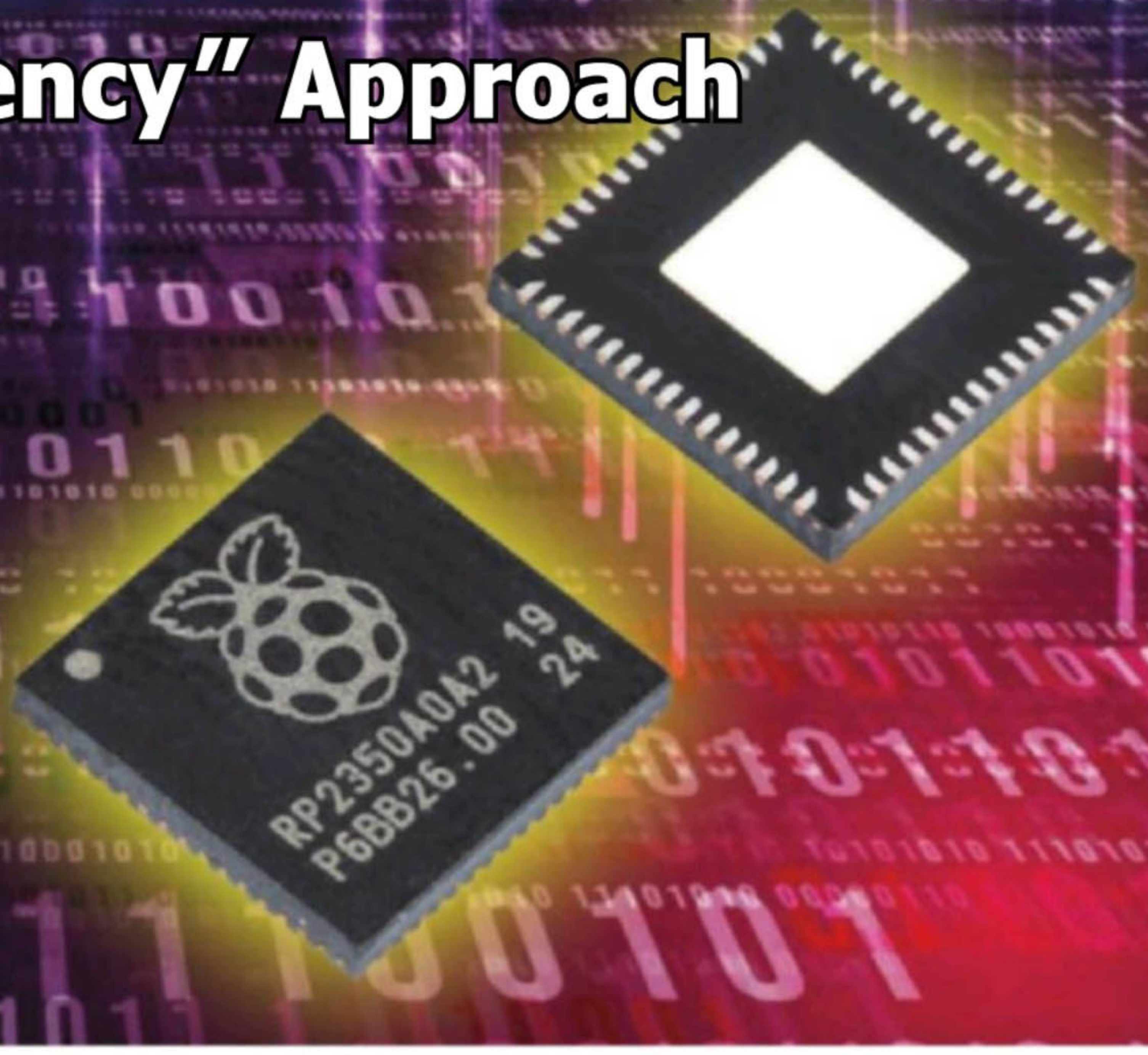
Register now and take your expertise to the next level!

Learn More: → **www.embeddedonlineconference.com**

Embedded System Essentials

Exploring the RP2350 Security

Raspberry Pi's Novel "Security through Transparency" Approach



By
Colin O'Flynn

Raspberry Pi's RP2350 MCU has more memory, faster processors, and lower power states compared to its predecessor, the RP2340. In this month's article, Colin explores some additional features. He emphasizes transparency, enhanced security with its innovative boot ROM and redundancy coprocessor, and glitch protection. He then shares the results of Raspberry Pi's "hacking challenge" to identify and fix security vulnerabilities in the chip.

Recently, Raspberry Pi introduced the RP2350 microcontroller, which has a huge amount of features under the hood. If you've used the older RP2040, you know it was already an interesting device, with unique features such as a high-speed "Programmable In Out" (PIO) block that allowed you to effectively bit bang high-speed protocols using a special-purpose peripheral block. But the RP2040 looks tame compared to the RP2350. One of the most ambitious features is that the entire architecture of the device can be changed. The RP2350 can be booted in two modes, a dual-core Cortex-M33 (ARM Cortex) mode, or a dual-core Hazard3 (RISC-V) mode.

One of the main features missing from the RP2040 was code security. The device booted off an external SPI flash, which meant that anybody could easily read out the flash memory. Making a secure system primarily meant using an external device that provided a secure boot, but adding an external device negated the low-cost advantage of the RP2040.

The RP2350 also fixes that problem, and as I'll discuss in this article, it introduces all sorts of novel features that haven't previously been available in a low-cost microcontroller. Unlike many "secure microcontrollers," Raspberrry Pi is keeping the public informed about challenges in their security, and has already published a series of blog posts about attacks on their first step of the microcontroller. At the end of the article, I take a look at known attacks on the RP2350.

RP2350 SECURE BOOT

To begin, I'll describe the basic secure boot flow of the RP2350. When discussing this MCU, I also have to emphasize that one of the most unique features is just how open everything is. The source code for its boot ROM can be found on Raspberry Pi's GitHub page [1]. Normally we'd have to rely on the manufacturer to describe the boot flow, and we'd have to trust their description. But for this device, you can also look at how the boot ROM was designed, not just trust a vague description in the manual.

Fundamentally, the fact that the RP2350 includes One Time Programmable (OTP) memory is what makes the device able to be

```
s_varm_step_safe_reset_unreset_block_wait_noinline:
   rcp_canary_get ip, CTAG_S_VARM_UNRESET_BLOCK_WAIT_NOINLINE

   ldr r1, =RESETS_BASE + REG_ALIAS_SET_BITS
   str r0, [r1, #RESETS_RESET_OFFSET]
   ldr r1, =RESETS_BASE + REG_ALIAS_CLR_BITS
   str r0, [r1, #RESETS_RESET_OFFSET]
   // Remove alias bits (note we’re avoiding v8-M Main instructions here)
   lsrs r1, #14
   lsls r1, #14
1:
   ldr r2, [r1, #RESETS_RESET_DONE_OFFSET]
   bics r0, r2
   bne 1b

   rcp_canary_check ip, CTAG_S_VARM_UNRESET_BLOCK_WAIT_NOINLINE
   bx lr
```

LISTING 1
A block of code that includes instructions to catch attacks.

secured. The existence of OTP alone does not mean a device is secure, but it can form the basis for a secure boot.

The boot ROM of the RP2350 uses the OTP to allow you to force it to boot authorized images. This can be configured to force images to pass a signature check, for example. Additionally, the configuration offers more advanced features, such as anti-rollback—which allows you to ensure someone cannot make the device boot an “old” firmware image after you already fixed security vulnerabilities.

Of course, such boot ROMs would normally be vulnerable to a “fault injection attack.” I covered this in previous articles, but I point you especially to my September 2018 article in *Circuit Cellar*, “Embedded System Essentials. Recreating Code Protection Bypass: An LPC MCU Attack” [2].

As an example of some of these features, you can see a block of code in **Listing 1,** which I’ve taken from the `arm8_bootrom_rt0.s` function of the boot ROM. It includes some instructions that aren’t your normal Arm assembly: `rcp_canary_get` and `rcp_canary_check`. These instructions are used to catch attacks where the function returns (with the `bx lr` on the final line of Listing 1), without having originally entered this function. This will catch a few types of typical attacks:. One of them would be Return Oriented Programming (ROP), where you corrupt the stack to build your own program, by using the return function in each subroutine to jump to different sections of code. If you jump to the middle of a protected subroutine, when the `rcp_canary_check` is done before the return, the RP2350 will throw an exception instead of allowing the return (`bx lr`) to occur.

But how exactly is it doing this, and what are these “RCP” instructions? The answer is that there is a special block called the Redundancy Coprocessor (RCP) that adds a lot of security into a special hardware block. Lets look at this in more detail.

REDUNDANCY COPROCESSOR (RCP)

The RCP is a unique block in the RP2350 that makes it easier for you to write more secure code. As I mentioned as one example, it has an ability to check that program flow is proceeding as expected. But it has a lot more capabilities as well, so I want to talk in more detail about the RCP and why you should find it especially interesting.

To begin, the RCP requires some random number to ensure you have unpredictable values on every program run, called the “salt register.” For the example of the stack canary commands in Listing 1, those commands write known-but-random values to the stack. In this case, “known but random” means it’s a value based on the fixed (known) value of the function ID that you define, and the random salt value. The RCP can check that the expected value is present on the stack before returning from a function, and if not, an unmaskable exception occurs.

But this is just one feature of the RCP. To introduce the rest, I want to bring back some code I’ve used previously, which is shown in

```
if (check_signature(&firmware_image) == 0) {
  while(1){
      ;//Loop forever since we cannot boot
  }
} else {
  do_boot(&firmware_image);
}
```

LISTING 2
Example of an unsecure bootloader implementation.

Listing 2. This was an example of an unsecure bootloader implementation. This pseudo-code (which looks like C) is designed to check that a firmware image variable (stored in `firmware_image`) has an acceptable signature. If it does, it boots the image, and if not, it blocks the boot and waits for the user to reset the system and load a new image. This sort of logic is used in many real bootloaders, but I'm avoiding picking on any vendor here.

```
rcp_count_set(5);
rcp_count_check(5);
if (check_signature(&firmware_image) == 0) {
  rcp_count_check(6);
  while(1){
      ;//Loop forever since we cannot boot
  }
} else {
  rcp_count_check(6);
  do_boot(&firmware_image);
}
```

LISTING 3
Validating the code flow.

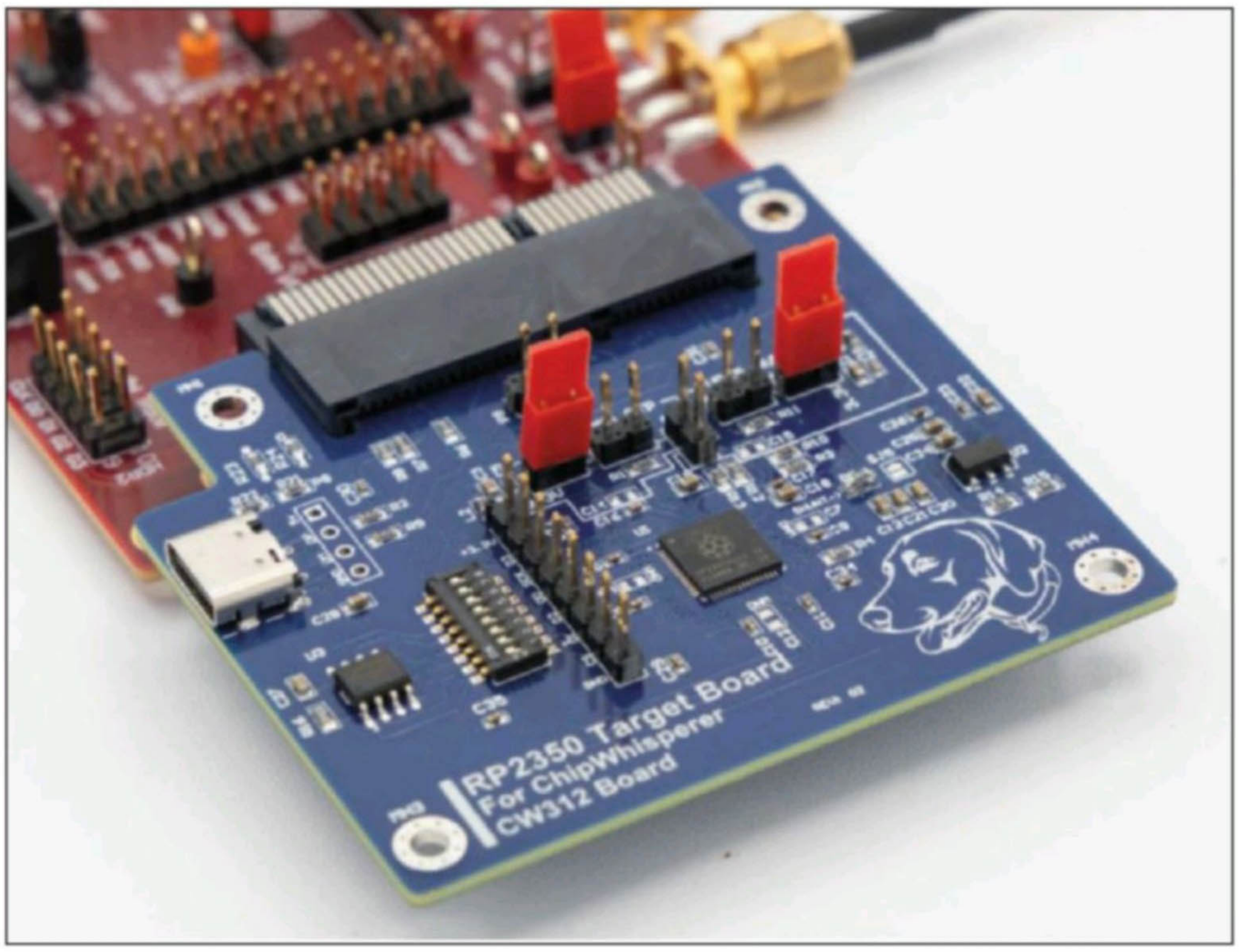


FIGURE 1
This target board includes features to test for side-channel power analysis and glitch protection.

It has a few problems, one of which is that, for example, using fault injection I can jump out of the `while(1)` loop and into the next branch, which has a call to `boot_image()`. This was an example of manipulating the control flow within a function, which stack canaries won't catch.

The RCP also allows us to add a check within the code flow, to validate that the expected control flow occurred, as shown in **Listing 3**. This works by initializing a special counter that is incremented throughout the code. At the same time it's incremented, the hardware validates whether the number on the RCP counter matches the expected number from the source code. So, in my example, if both code flows are executed, it would mean there would be one more count than expected, causing a mismatch (since the correct code flow executes one or the other branches of the `if()` statement, and my attacker code-flow executed both).

The RCP also helps with the problem of checking and storing values. Say, for example, that my simple Boolean response from the `check_signature()` call in Listing 2 has the problem that if the value is corrupted to any non-zero value, the wrong path is taken. Instead, these Booleans can be encoded as more complex values that are checked by the RCP, such that if the value is not one of two valid magic values, it again causes an exception. Additionally, the RCP can store integers as special values that are redundantly stored, and it can even perform various tests on different values.

The final key feature of the RCP is that instructions add random delays into the execution path. This makes performing fault injection much more difficult, since every execution will have slightly different timing. Typically, I'm exploiting highly consistent boot timings, when looking at attacks on devices.

FAULT INJECTION & SIDE CHANNEL TESTING

While the RCP sounds good in theory, how do we go about testing it? To help with testing this chip, we designed a special target board shown in **Figure 1**. This board includes a lot of features for testing both side-channel power analysis and glitch protection on this device. The design is available from an open-source repository with other ChipWhisperer targets [3]. One additional protection against glitches is a glitch detector. Back in my May 2020 article in *Circuit Cellar*, "Embedded System Essentials. Broad Market Secure MCUs: Spotlight on the MAX32520" [4], I ran some tests on a glitch detector on another

Additional materials from the author are available at:
www.circuitcellar.com/article-materials
References [1] to [6] as marked in the article can be found there.

Resources
ChipWhisperer | www.newae.com/chipwhisperer
IOActive | www.ioactive.com
Raspberry Pi | www.raspberrypi.com

embeddedworld
Exhibition&Conference

CONNECTING THE
EMBEDDED COMMUNITY
11.–13.3.2025
NÜRNBERG, GERMANY

Get your
free ticket now!
embedded-world.de/code
Use the voucher code **GGew25**

Medienpartner

elektroniknet.de
Markt&Technik
Elektronik
Elektronik automotive
Elektronik medical
connect professional

NÜRNBERG MESSE

device—which had limited success detecting the glitch. By comparison, the RP2350 glitch detector can be easily triggered; this is a good thing, meaning it is successful in detecting the glitches. But you can explore the glitch detector easily to understand how it's used with this platform.

As you can expect, it's hard to get a design perfect the first time, and a few exploits were indeed found on the RP2350, as a result of a hacking challenge issued by Raspberry Pi [5].

SECURITY THROUGH TRANSPARENCY

There is a common phrase, "security through obscurity," meaning that we try to avoid letting a would-be attacker know things about our device. This is usually the norm in the competitive microcontroller market, where you'd have to sign an NDA to learn about even the existence of attacks against the device.

In contrast, Raspberry Pi ran a hacking competition [5] that resulted in several attacks against it. I'll briefly summarize them here, but check out the full blog post for details—including links to code and demonstrations [6].

The first attack by Aedan Cullen took advantage of the dual-boot mode of the RP2350. Using a voltage glitch, it was possibly to bypass the expected boot mode, and boot into RISC-V mode with debug enabled. This is related to how the OTP memory worked; it was possible to glitch this read to cause the device to come up in an unexpected mode.

The second attack by Marius Muench is related to the USB bootloader. While the fault injection mitigations I mentioned earlier are part of the bootloader, there was also a path that uses unsanitized input data which, in combination with a voltage fault injection attack, results in the device booting code that an attacker previously loaded into RAM. Luckily, there is a OTP setting to disable the feature required for this attack, so it's less critical than the one by Cullen.

A low-cost laser fault injection setup was used by Kévin Courdesses to cause the hash function to calculate the hash over the wrong data. This means an attacker can cause the signature check to pass, because the firmware being checked isn't the one that is actually about to be run! This is interesting, because it bypasses the glitch detectors using a laser fault injection platform, and it's incredibly impressive to see such attacks working in practice!

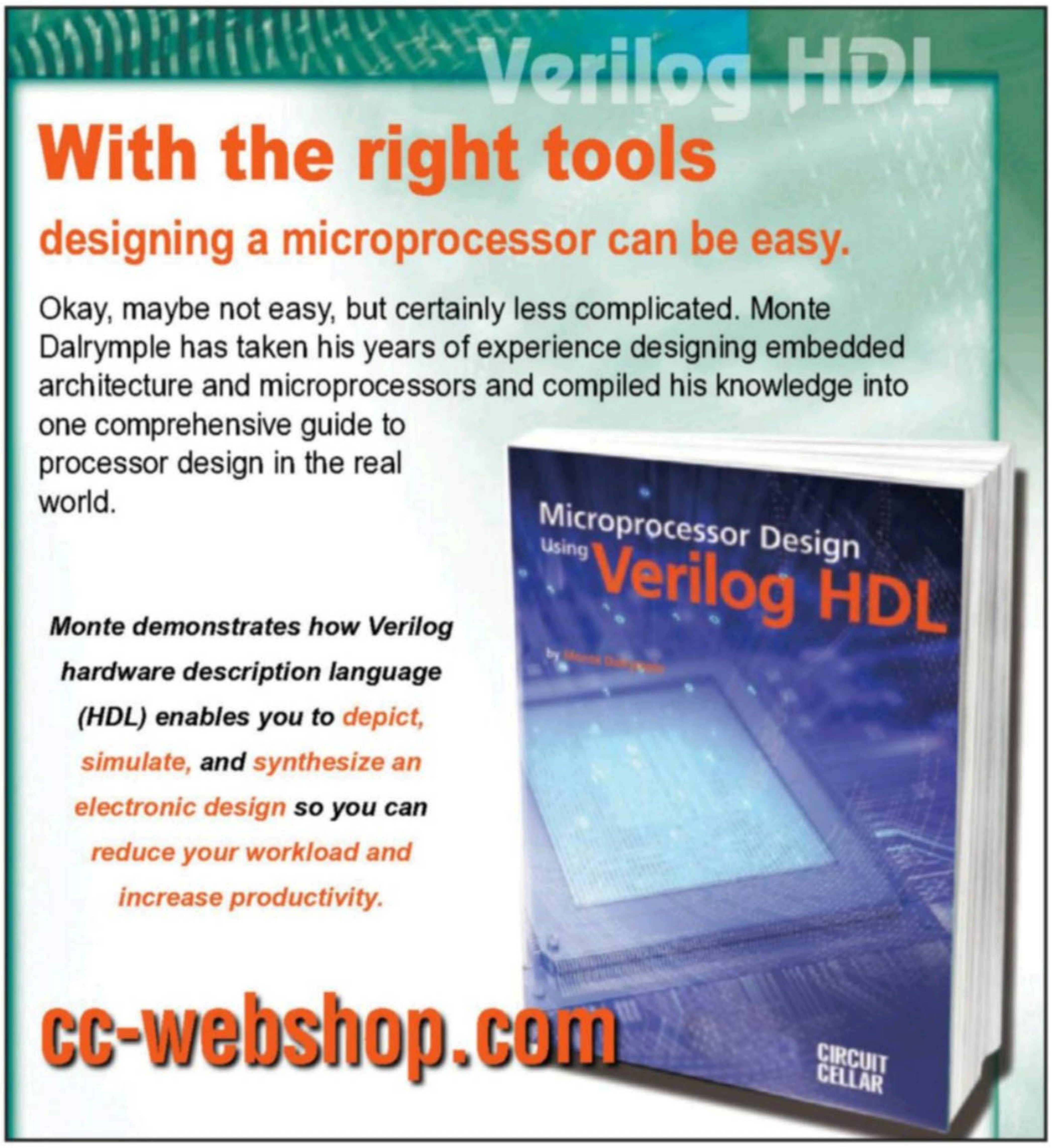
At the higher end, IOActive used a much more advanced technique. A focused ion beam (FIB) was used to read out secret memory. Although it may seem like a high-budget technique, it might make sense in many situations. Recovery of bitcoin wallets is one that comes to mind, for example, but there are lots of high-value products where this could be in-scope.

Finally, Thomas Roth reported several interesting results, including a double-glitch attack that bypasses the OTP protection, allowing further reads and writes to an OTP page, when it should not be possible. Other evaluation results are likely to be released in the future, so keep an eye out for what has been done with the RP2350 device.

I hope you enjoyed this look at what I think will be the future of secure IoT microcontrollers—I think it will be the future not because of the specific piece of silicon, but the future due to the transparency built around the product.

ABOUT THE AUTHOR

Colin O'Flynn (colin@oflynn.com) has been building and breaking electronic devices for many years. He is CTO of NewAE Technology based in Halifax, NS, Canada and is currently an assistant professor at Dalhousie University. Some of his work is posted on his website at www.colinoflynn.com.



Verilog HDL

With the right tools

designing a microprocessor can be easy.

Okay, maybe not easy, but certainly less complicated. Monte Dalrymple has taken his years of experience designing embedded architecture and microprocessors and compiled his knowledge into one comprehensive guide to processor design in the real world.

Monte demonstrates how Verilog hardware description language (HDL) enables you to depict, simulate, and synthesize an electronic design so you can reduce your workload and increase productivity.

cc-webshop.com

By Kirsten Campbell **PRODUCT NEWS**

IAR Expands Support for TMC THA6 Gen2 Series Automotive MCUs

IAR, a global leader in embedded development solutions, announces its support for the THA6 Gen2 series automotive-grade MCUs from Tongxin Microelectronics Co., Ltd. (TMC). This collaboration, achieved through the latest release of IAR Embedded Workbench for ARM v9.60.3, enables developers to meet next-generation automotive electronics' high-performance and functional safety demands.

The TMC THA6 Gen2 series is the first ARM Cortex-R52+ ASIL D MCU in China. Operating at a frequency of 400MHz, it is built on the modern Arm v8 architecture, offering both performance and scalability. Additionally, it integrates the latest GTM 4.1 (Generic Timer Module), supports high-resolution Pulse Width Modulation (PWM), and complies with EVITA-Full, a standard for automotive cybersecurity ensuring secure communication and encryption. Featuring a built-in hardware Resolver-to-Digital Converter (RDC) module, the chip supports both software and hardware decoding of resolver signals, which are critical for motor position sensing. This combination provides a reliable foundation for Electrical/Electronic architectures (E/E) architectures, addressing the demands of applications such as powertrain, chassis, body systems, and intelligent driving. By integrating IAR's advanced development tools, this collaboration streamlines the creation of safe and high-performance automotive systems, reducing manufacturers' time to market.

IAR Embedded Workbench for ARM Functional Safety Edition is certified by TÜV SÜD and supports ISO 26262 ASIL D compliance, making it an industry benchmark for functional safety. Developers benefit from a robust toolchain that includes:

- Highly optimized code generation: Ensuring compact and high-performance applications.
- Integrated debugging tools: Simplifying the identification and resolution of complex issues.
- Advanced analysis capabilities: Tools like C-STAT for static code analysis and C-RUN for runtime analysis help ensure robust code quality and early detection of potential issues.

These features empower developers to fully leverage the TMC THA6 series, ensuring system compliance with the automotive industry's stringent safety standards.

IAR | iar.com

Power Integrations' Latest MotorXpert Software Drives FOC Motors Without Shunts or Sensors

Power Integrations, the leader in high-voltage integrated circuits for energy-efficient power conversion, announced MotorXpert v3.0, a software suite for configuration, control, and sensing of BLDC inverters that utilize the company's BridgeSwitch motor-driver ICs. The latest release of the software incorporates Power Integrations' shuntless and sensorless technology for field-oriented control (FOC), adding support for advanced modulation schemes and unconditional startup under any load condition, along with significant improvements to the host user interface and debugging tools.

MotorXpert 3.0 comprises three main sections:

- Sophisticated mathematical algorithms resident on the local MCU or DSP construct accurate feedback signals from the BridgeSwitch IC and provide real-time control of the switching patterns.
- A host-side application interprets inverter actions and displays critical data in actionable format for engineering analysis.
- The easy-to-use control interface permits development engineers to experiment and quickly converge to a final product.

MotorXpert suite is MCU-agnostic and includes a porting guide to simplify deployment with a wide range of MCUs. It is implemented in common C language to MISRA standards.

MotorXpert Suite

The MotorXpert v3.0 host-side application includes a graphical user interface with Power Integrations' digital oscilloscope visualization tool that makes it easy to design and configure parameters and operation and simplifies debugging. Parameter tool tips and a tuning assistant improve the development process, and the intuitive parameter list provides easy motor tuning. The new version also features both V/F and I/F control, which permits motor startup in any load condition. A selectable two-phase modulation scheme allows developers to trade off temperature of the inverter vs. torque ripple, which is beneficial in applications such as hot water circulation pumps, reducing heat sinking requirements and enclosure cost. Development time is greatly reduced by the included single- and three-phase code libraries with sensorless support, reference designs, and other tools such as a power supply design and analysis tool.

The BridgeSwitch IC is a half-bridge motor driver of low RDSON FREDFET switches, controllers and drivers in a thermally efficient package. Sensorless feedback, fault reporting, and self and load protection are also available for applications from 30W to 1 HP (750W).

Applications include indoor and outdoor air conditioning fans, refrigerator compressors, fluid pumps, washing machine and dryer drums, range hoods, industrial fans, and heat pumps.

Power Integrations
power.com

NEW PRODUCT SUBMISSIONS— E-mail: product-editor@circuitcellar.com

PRODUCT NEWS *By Kirsten Campbell*

HSYCO Unveils New BACnet Server Driver for Enhanced Building Automation

HSYCO, a leading provider of integrated building management solutions, is excited to announce the release of its new BACnet Server driver. This latest addition further expands HSYCO's capabilities, enabling seamless integration between BACnet and the over 100 protocols it already supports.

With the BACnet Server driver, HSYCO acts as a powerful gateway, facilitating data exchange and interoperability between diverse systems, including building automation, industrial control, and IoT solutions. This enhances flexibility, scalability, and efficiency in managing complex automation environments.

Seamless Integration: The BACnet Server driver enables seamless communication between BACnet devices and other systems supported by HSYCO.

Enhanced Interoperability: HSYCO acts as a gateway, allowing data exchange and interoperability between diverse building automation, industrial control, and IoT solutions.

HSYCO Unveils New BACnet Server Driver for Enhanced Building Automation
HSYCO
BACnet

Improved Usability: The BACnet Server Utility simplifies the visualization of BACnet/IP device properties connected to HSYCO and streamlines the linking of BACnet attribute values to HSYCO datapoints.

Increased Flexibility: The ability to integrate BACnet devices with a wide range of other systems enhances the flexibility and adaptability of building automation solutions.

Enhanced Scalability: HSYCO's scalable architecture can handle complex building automation projects.

Improved Efficiency: Interoperability reduces duplicative activities and costs associated with managing disparate systems.

Enhanced Security: HSYCO's secure platform provides a safe and reliable environment for building automation operations.

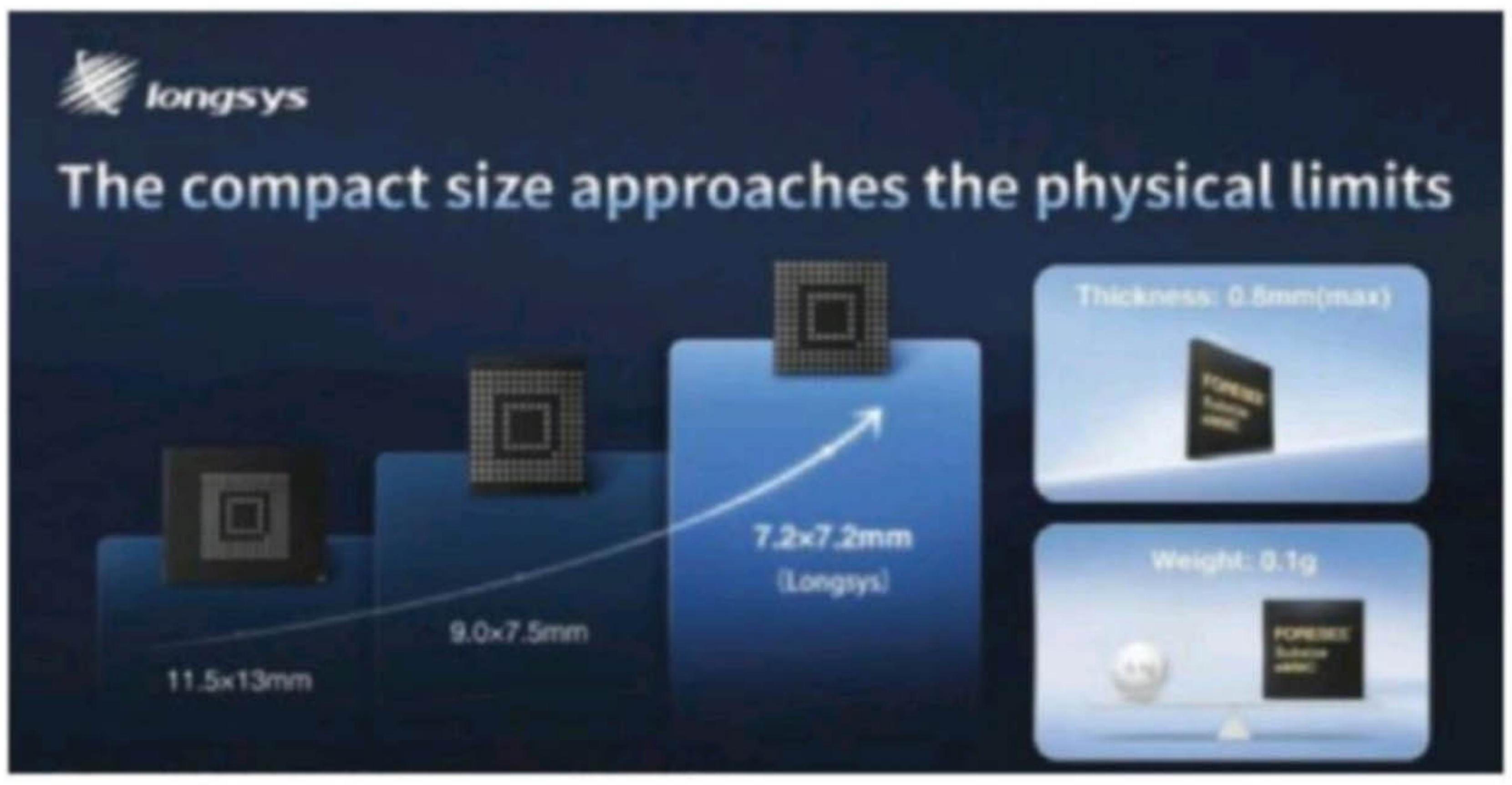
HSYCO | hsyco.com

Longsys Launches 7.2mm Subsize eMMC, Breaking the Physical Space Constraints of AI Wearables

In the design of wearable devices, every millimeter matters. As AI technology becomes deeply integrated, wearable devices demand not only enhanced performance but also the ability to deliver more functionalities within highly limited spaces. Recently, Longsys (301308.SZ) introduced a 7.2mm × 7.2mm subsize eMMC, providing a groundbreaking memory solution for optimizing the physical space of AI wearables.

Ultra-Compact: Unlocking New Possibilities for Wearable Design. 7.2mm × 7.2mm is one of the smallest subsize eMMCs currently available on the market, achieving maximum space efficiency. Its 153 solder balls nearly cover the entire panel, pushing the design to the very edge of physical limits. Compared to the standard 11.5mm × 13mm eMMC, its surface area is reduced by approximately 65%, with a thickness of just 0.8mm. Featuring a lightweight design, it weighs only 0.1g (approx.), nearly 67% lighter than the standard 0.3g eMMC. This ultra-compact design frees up additional space for other internal components, enabling wearable devices to maintain a sleek and lightweight form while integrating more functional modules to meet diverse user demands.

While pursuing an ultra-compact design, the 7.2mm × 7.2mm subsize eMMC does not compromise on performance or capacity. Equipped with in-house firmware, it ensures fast device boot-up, seamless AI application performance, and efficient data processing. Additionally, low-power technologies, such as intelligent sleep and dynamic frequency scaling, significantly reduce energy consumption and extend battery life without compromising performance. With capacities of 64GB and 128GB—among the highest for similar compact products—it offers innovative solutions for optimizing AI wearables such as smart glasses, watches, and earphones.



In-House Packaging and Testing: Leading the Industry

Notably, this subsize eMMC is packaged and tested at Longsys' in-house Suzhou Packaging & Testing Base, and it adopts an innovative grinding and cutting process to achieve a smaller size. The facility specializes in NAND Flash and DRAM packaging and testing while extending its capabilities to eMMC, UFS, eMCP, and ePOP series, offering comprehensive services across wafer-level, chip-level, and system-level packaging.

Beyond Compactness, advanced packaging technologies give Longsys a competitive edge in product size, thermal management, compatibility, reliability, and storage capacity. Longsys' eMMC products are now highly mature, with 2024 seeing the launch of its proprietary eMMC controllers and QLC eMMC. Beyond compact design, its full-stack customization capabilities extend across product development, technology, and manufacturing to meet diverse client needs.

The launch of the 7.2mm × 7.2mm ultra-compact eMMC represents a new breakthrough for mature products, offering robust support and enhanced options for AI wearable development. Looking ahead, Longsys remains committed to memory technology innovation, bringing more exciting possibilities to wearable devices.

Longsys | longsys.com

NEW PRODUCT SUBMISSIONS— E-mail: product-editor@circuitcellar.com

Pison and STMicroelectronics Announce Timex as Neural Sensor Partner, Ushering a New Era of Neurocognitive Wearables

Pison, the trailblazer in AI-powered neural sensors for neurocognitive health, wellness, and gesture control, announced it has begun licensing its innovative neural sensor to leading smartwatch and wearable brands.

The company worked with STMicroelectronics (ST), a global semiconductor leader serving customers across the spectrum of electronics applications, to miniaturize and reduce the cost of the sensor to fit in a consumer watch form factor. Pison is now licensing brand OEMs and ODMs to integrate ST's ENG-enabled biosensor chips and Pison's natural language gesture control stack into their watches and wearables.

"Pison's electroneurography technology offers a major leap forward for wearables. Its ability to accurately measure brain activity from a wrist-worn wearable opens up a whole new class of applications and usage models," said Marco Angelici, VP of Marketing and Applications for Analog and MEMS at STMicroelectronics. "We're excited to collaborate with Pison to bring this cutting-edge technology to market and echo their vision to become ubiquitous in a new generation of smartwatches and athletic wearables."

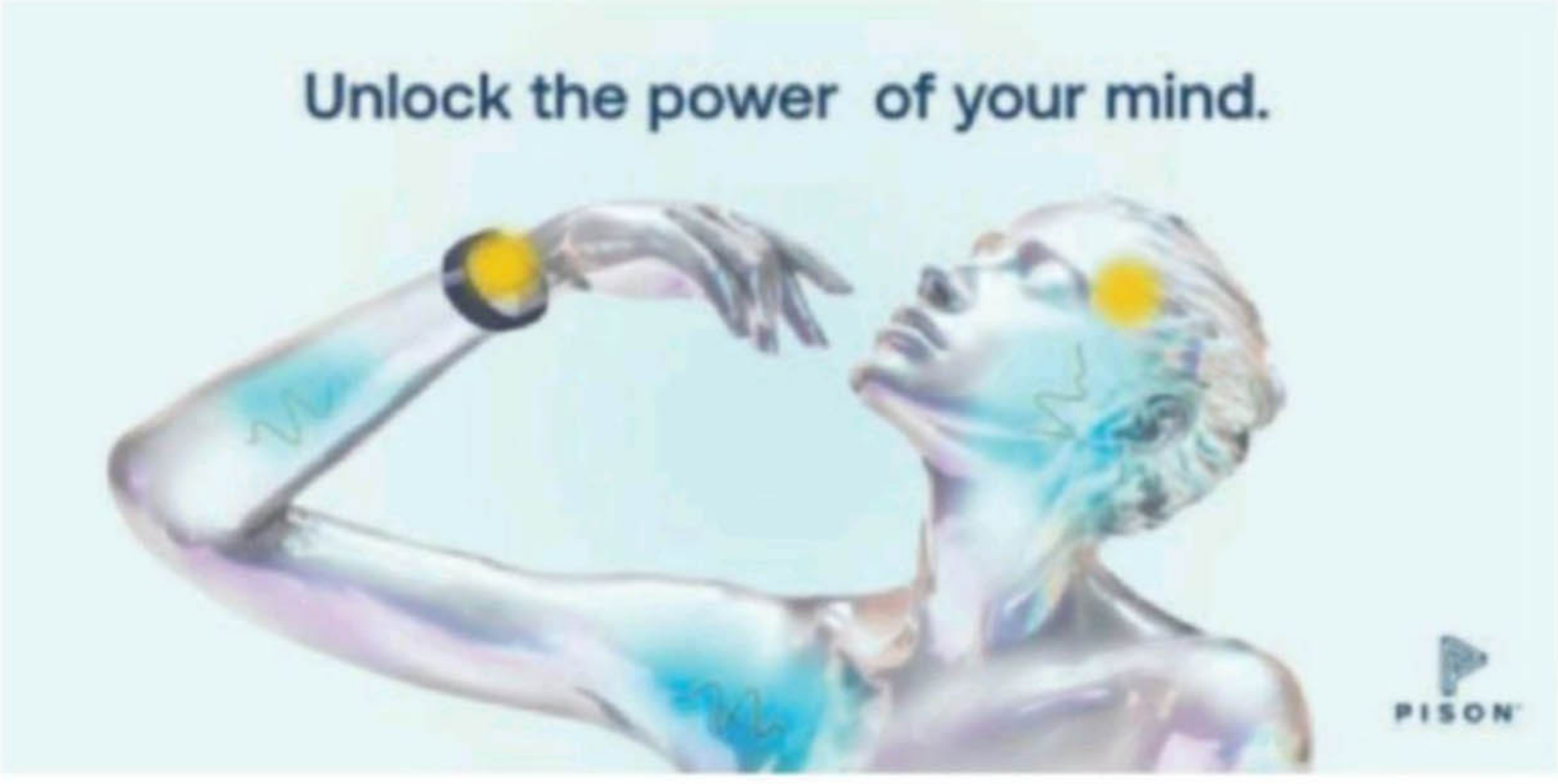
STMicroelectronics and Pison worked together to integrate the company's core sensor library into Qualcomm's Snapdragon W5 platform. Pison now operates as an authorized ISV enabling watch OEMs and ODMs in the Snapdragon ecosystem.

In related news, Pison announced last week that Samsung Ventures has invested in an active equity round. Pison is now poised and fully funded to launch a new generation of mental health, wellness, and performance apps with subscription services that make use of this neural sensor in ways that will change our world.

Timex plans to develop the next generation of neurocognitive wearable devices with Pison. In addition, a variety of Timex-branded devices "powered by Pison" will start to roll out to commercial marketplaces and consumers in the spring.

Pison Apps and ENG-enabled smartwatches is expected to deliver major societal impacts in mental health and wellness, athletic excellence, workforce safety, neurological health, brain health, and natural language gesture control.

Pison | pison.com


Unlock the power of your mind.
PISON

Wearable Devices Announces Availability of its Mudra Link Neural Gesture-Control Wristband

Wearable Devices, Ltd., announced the general availability of its Mudra Link, the first neural wristband for Android, macOS, and Windows devices. This achievement marks the company's continued expansion as the industry pioneer in wearable gesture control, following the new device's CES 2025 Innovation Awards recognition in the XR Technologies & Accessories category.

Together with the company's Mudra Band for Apple devices, itself a CES Best of Innovation award-winner, the Mudra Link represents a new era of touchless control and interaction. The Mudra's proprietary Surface Nerve Conductance sensors pick up electromyography signals from subtle finger movements, translating them into intuitive commands for a wide range of devices and applications, including augmented reality (AR) glasses, smart TV streamers, mobile phone, tablet and personal computers, and smart home control.

"The Mudra Link reflects Wearable Devices' commitment to advancing neural interface technology for a seamless and practical user experience," said Asher Dahan, Chief Executive Officer of Wearable Devices. "As the first neural gesture control device compatible with all the leading operating systems platforms, the Mudra Link reinforces our role as pioneers in redefining human-machine interaction."

mudra
LINK
The Future is Here!
ENDLESS POSSIBILITIES
AT YOUR FINGERTIPS.

Complementing XR Advancements, Including Meta's Orion Glasses. As AR, virtual reality (VR), and extended reality (XR) technologies continue to evolve, the Mudra technology is positioned as a critical enabler for extended reality ecosystems. Studies and insights, including research from Meta, suggest that devices like Orion AR glasses are enhanced by neural gesture control wearables like Mudra Band and Mudra Link, extending user interaction well beyond visual fields and physical interfaces.

Intuitive Gesture Control: With advanced neural interface technology, Mudra Link captures the smallest movements of a user's hand and translates them into precise commands.

Customizable Gestures: Users can make Mudra Link their own by personalizing gestures for their favorite apps, giving them total control over their digital life. It enables the mapping of gestures to specific commands to create customized interactions.

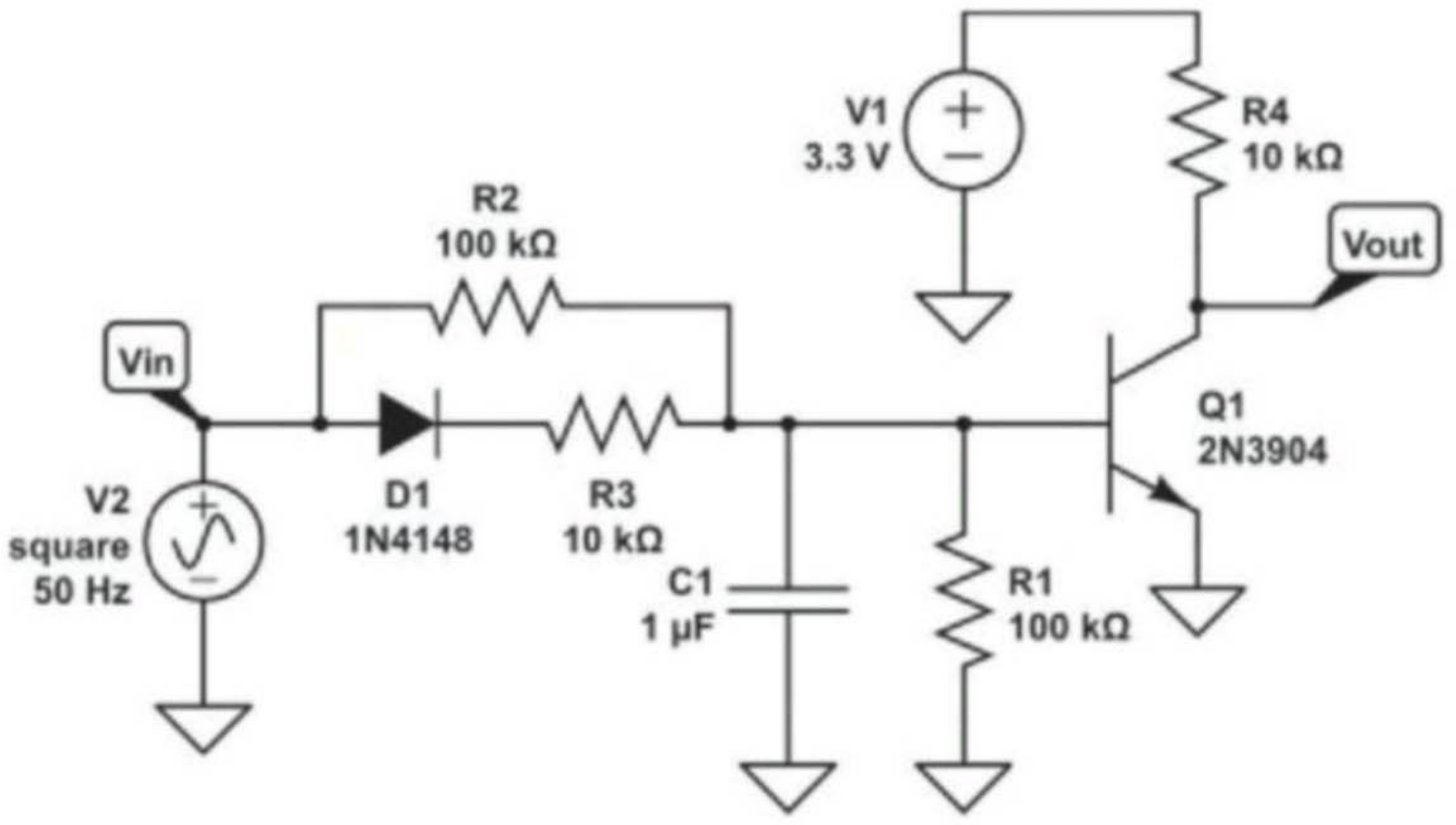
Compatible with Everything: Whether using iOS, Android, Windows, or macOS, Mudra Link works across platforms, opening up new possibilities for gaming, smart home automation, and even professional tasks.

Wearable Devices
wearabledevices.co.il

TEST YOUR EQ

Contributed by David Tweed

Question 1 —
A circuit is required that grounds its output with a certain delay after the logic-level input changes state, and releases that ground after a different, longer delay. The following circuit is proposed. What is the delay on the low-to-high transition of the input?


V1
3.3 V
R4
10 kΩ
R2
100 kΩ
Vout
Vin
Q1
2N3904
V2
square
50 Hz
D1
1N4148
R3
10 kΩ
C1
1 µF
R1
100 kΩ

Question 2 —
What is the delay on the high-to-low transition of the input?

Question 3 —
What is the fundamental problem with this design?

Question 4 —
What kind of changes would make this a more predictable circuit?

POWERING INNOVATION:
DISCOVER THE R24C2T25 –
THE NEXT GENERATION IN
DC/DC CONVERSION

SEAMLESSLY INTEGRATE CUTTING-EDGE EFFICIENCY WITH OUR 2W
ISOLATED SMD CONVERTER FOR ADVANCED IGBT/SIC GATE DRIVERS

RECOM
R24C2T25 series
12.83 mm
7.5 mm
3.18 mm

RECOM
recom-power.com/gate-drivers

IDEA BOX

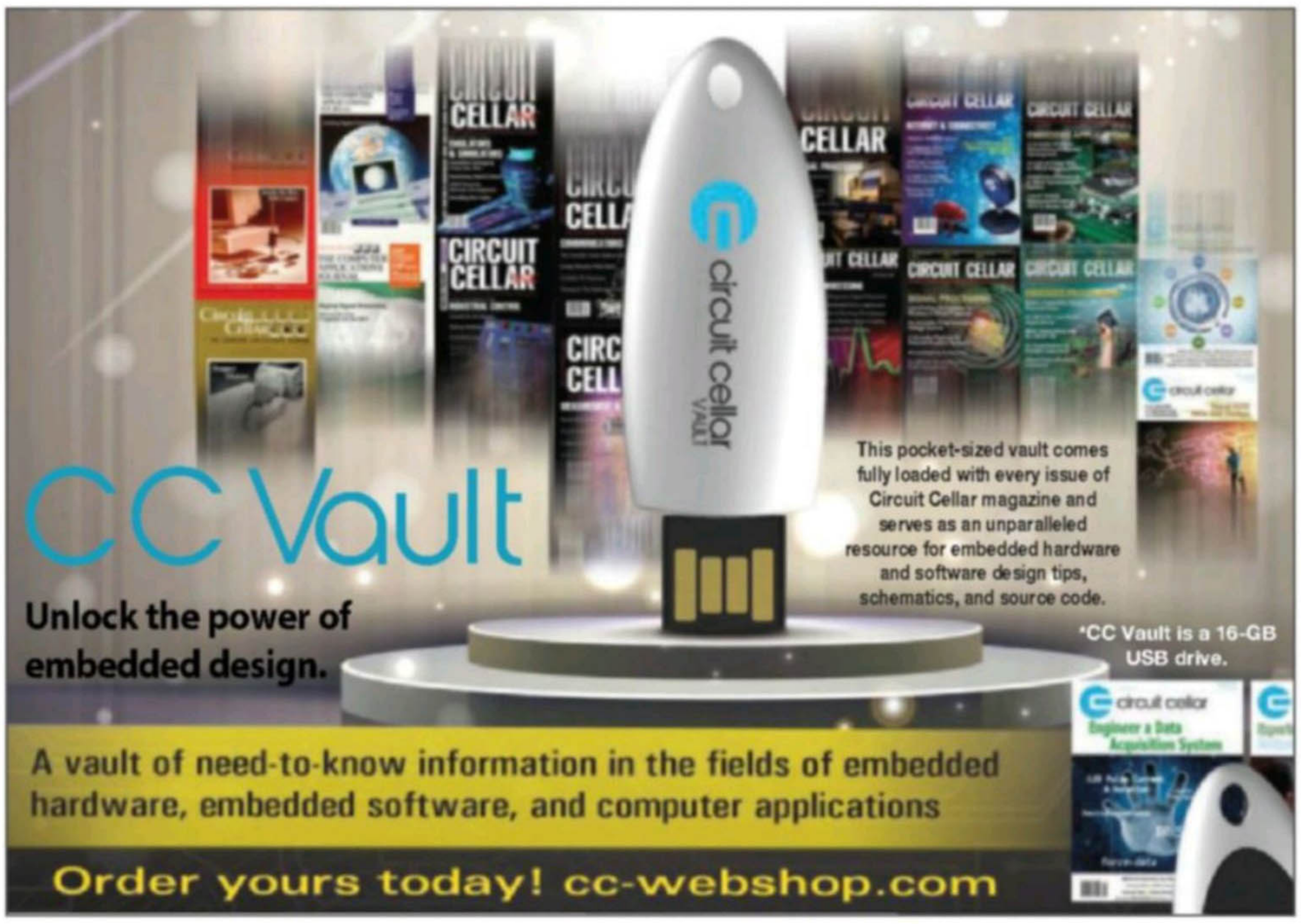
The Directory of PRODUCTS & SERVICES

AD FORMAT:

Advertisers must furnish digital files that meet our specifications (www.circuitcellar.com/mediakit).

All text and other elements MUST fit within a 2" x 3" format.
E-mail adcopy@circuitcellar.com with your file.

For current rates, deadlines, and more information contact Hugh Heinsohn at 757-525-3677 or Hugh@circuitcellar.com



CC Vault

Unlock the power of embedded design.

This pocket-sized vault comes fully loaded with every issue of Circuit Cellar magazine and serves as an unparalleled resource for embedded hardware and software design tips, schematics, and source code.

*CC Vault is a 16-GB USB drive.

A vault of need-to-know information in the fields of embedded hardware, embedded software, and computer applications

Order yours today! cc-webshop.com

RAPID18

Universal Development Kit for PIC18 Family

ONLY $99!

- Real-time Clock/Calendar with Supercap
- USB port for text communication to the running program
- Built-in bootloader for loading code
- Excellent for AC/DC Data Logging

Kit includes everything you need to develop with a PIC® MCU:

- Rapid 18 Prototyping Board
- Single-Chip IDE C Compiler
- Exercise Book
- Power Adapters and Cables

sales@ccsinfo.com (262) 522-6500 x35
www.ccsinfo.com/cc1223

PIC® MCU and dsPIC® are registered trademarks of Microchip Technology Inc.

When it comes to robotics, the future is now!

Advanced Control Robotics simplifies the theory and best practices of advanced robot technologies, making it ideal reading for beginners and experts alike.

With this book, you'll learn about:

- *Communication*
- *Technologies*
- *Control Robotics*
- *Embedded Technology*
- *Programming Language*
- *Visual Debugging...*

and more

Get it today, cc-webshop.com

ADVANCED CONTROL ROBOTICS
HANNO SANDER
start
start task async
repeat
start task sync and wait
task async
repeat
print text
wait 1000
task sync
print text
wait 1000
circuit cellar
Elektor

www.embeddedTS.com

TS-7100

- NXP i.MX 6UL Arm Cortex®-A7 @ 696 MHz
- 512 MB RAM
- 4 GB eMMC Flash Storage
- Ethernet, USB, RS-232, RS-485, and CAN
- Dual Ethernet, WiFi, Bluetooth
- NimbeLink/Digi socket

embeddedTS

The Future of Voice-Enabled AI

Industry Advancements in MEMS Microphones for Voice AI

By
Uday Mudoi
Vice President & General Manager - Microphone Business Unit, InvenSense, a TDK Group Company

Acoustic Activity Detection and High Acoustic Overload Point Transform Voice Interaction

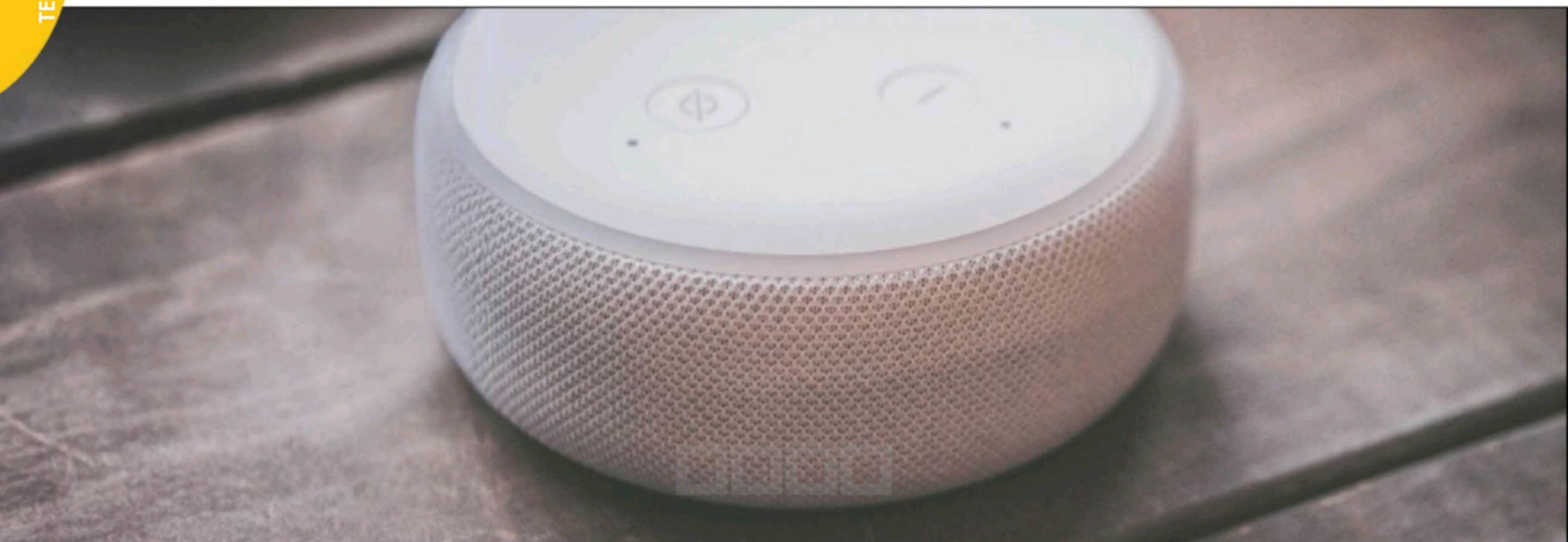
Advancements in MEMS microphones are enhancing voice-enabled AI technologies across various consumer electronics. Advancements in high signal-to-noise ratio and low power consumption enable more accurate and efficient voice recognition in devices ranging from smart speakers to wearables.

The rise of artificial intelligence (AI), especially in generative AI and natural language processing, is reshaping the consumer electronics landscape. Voice-controlled systems are becoming a key component of our everyday interactions with technology, from smart home devices to automotive interfaces. At the heart of this revolution are micro-electro-mechanical systems (MEMS) microphones, essential for capturing clear, accurate sound in various environments.

As AI technologies evolve, MEMS microphones play an increasingly critical role in enabling more user-friendly voice interfaces. The latest generation of MEMS microphones offer the high signal-to-noise ratio (SNR) and low power consumption necessary to meet the demands of today's always-on, voice-activated systems.

ELEVATING AUDIO QUALITY

One of the most essential aspects of AI-driven devices is their ability to capture sound accurately, regardless of background noise or environmental conditions. MEMS microphones are uniquely equipped to provide this level of sound capture, offering studio-quality audio that can rival traditional microphone systems. This reliability is particularly important for voice-activated assistants, smart home systems, and



wearables, where sound clarity is essential for effective performance.

One of the key features of MEMS microphones is their ability to operate on ultra-low power, making them ideal for battery-powered devices like wireless earbuds and smartwatches, as shown in **Figure 1**. These microphones provide the high SNR needed for clear voice recognition while consuming minimal energy, ensuring that the device can remain operational without frequent recharging.

In addition to providing high-quality audio, some MEMS microphones offer a high acoustic overload point (AOP), allowing them to capture sound accurately even in noisy environments. This makes them suitable for a wide range of use cases, from noise-canceling headphones to AI-driven robotics systems.

ESSENTIAL POWER EFFICIENCY

As the number of always-on, voice-activated devices continues to grow, power efficiency has become a top priority for manufacturers. Devices such as smart speakers, home assistants, and IoT systems must be able to listen for voice commands at all times without draining battery life. This is where MEMS microphones with advanced power-saving technologies come into play.

One of the most significant innovations in this area is Acoustic Activity Detection (AAD) technology, which allows MEMS microphones with AAD feature to remain in a low-power standby mode until they detect sound. This enables the microphone to conserve energy when no voice commands are being issued, extending the device's battery life. AAD technology is a game-changer for battery-powered devices like wireless headphones or smart home sensors, allowing them to remain permanently on without sacrificing performance.

HIGH AOP FOR NOISY ENVIRONMENT

Many consumer devices struggle to capture sound accurately in noisy environments. Whether it's urban streets, crowded airports, or factory floors, these environments present unique challenges for voice recognition systems. MEMS microphones with a high AOP are designed to overcome these challenges by capturing sound without distortion, even at high volumes.

For example, in noise-canceling headphones, a high AOP allows the microphone to accurately capture both background noise and the user's voice, enabling the system to cancel out unwanted sounds while maintaining clear voice input. This technology is also essential for AI-driven systems like robotic assistants and smart home devices, where the ability to operate effectively in dynamic environments is crucial.

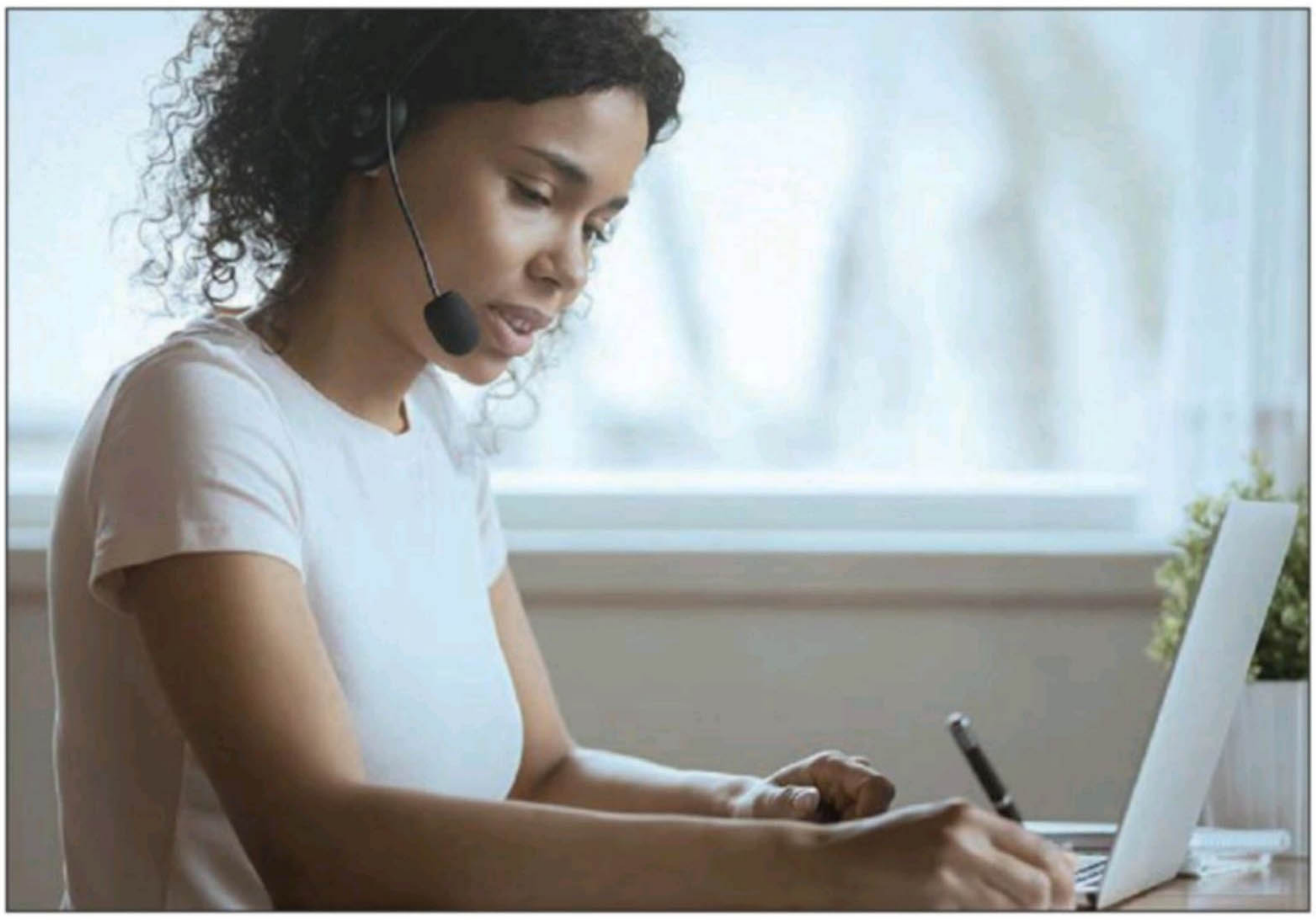


FIGURE 1
MEMS microphones operate on ultra-low power, making them ideal for battery-powered devices like Bluetooth headsets, wireless earbuds and smartwatches. (Image source: TDK IvenSense)

DIGITAL INTERFACE SIMPLIFIES DESIGN

Along with providing high-quality sound and power efficiency, the advanced MEMS microphones offer a range of digital interfaces that simplify system design and reduce the need for additional processing hardware. Pulse-Density Modulation (PDM) and Inter-IC Sound (I^{2}S) interfaces allow more efficient communication between the microphone and the device's processor, improving overall performance.

For example, the I^{2}S interface reduces the need for system developers to implement

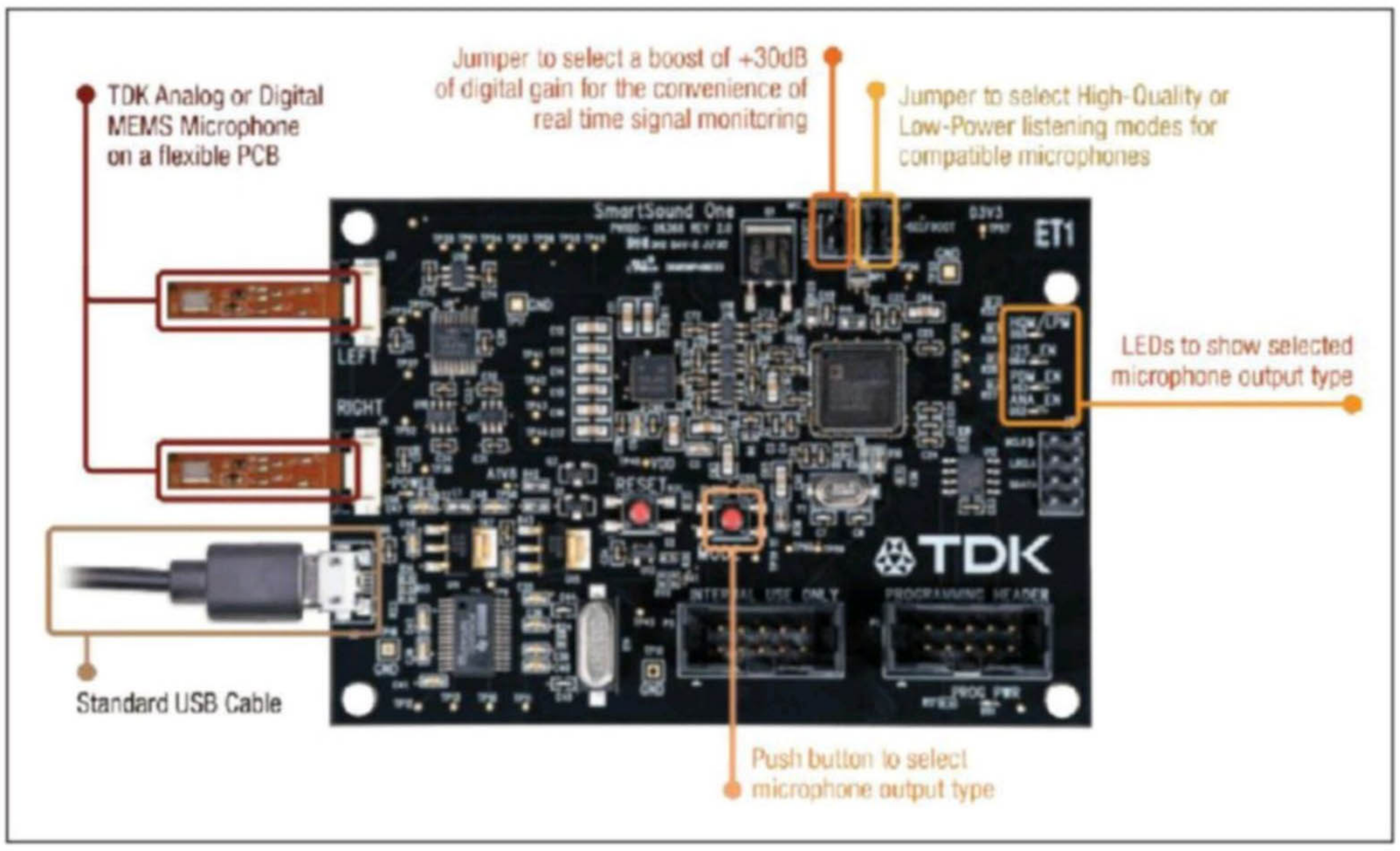

TDK Analog or Digital
MEMS Microphone
on a flexible PCB
Jumper to select a boost of +30dB
of digital gain for the convenience of
real time signal monitoring
Jumper to select High-Quality or
Low-Power listening modes for
compatible microphones
SmartSound One
ET1
LEFT
RIGHT
RESET
LEDs to show selected
microphone output type
TDK
INTERNAL USE ONLY
PROGRAMMING HEADER
Standard USB Cable
Push button to select
microphone output type

FIGURE 2
The SmartSound One, a USB plug-and-play microphone interface module, is compatible with PDM digital, I^{2}S digital, single-ended analog, and differential analog TDK microphones supplied on a flexible PCB. (Image source: TDK InvenSense)

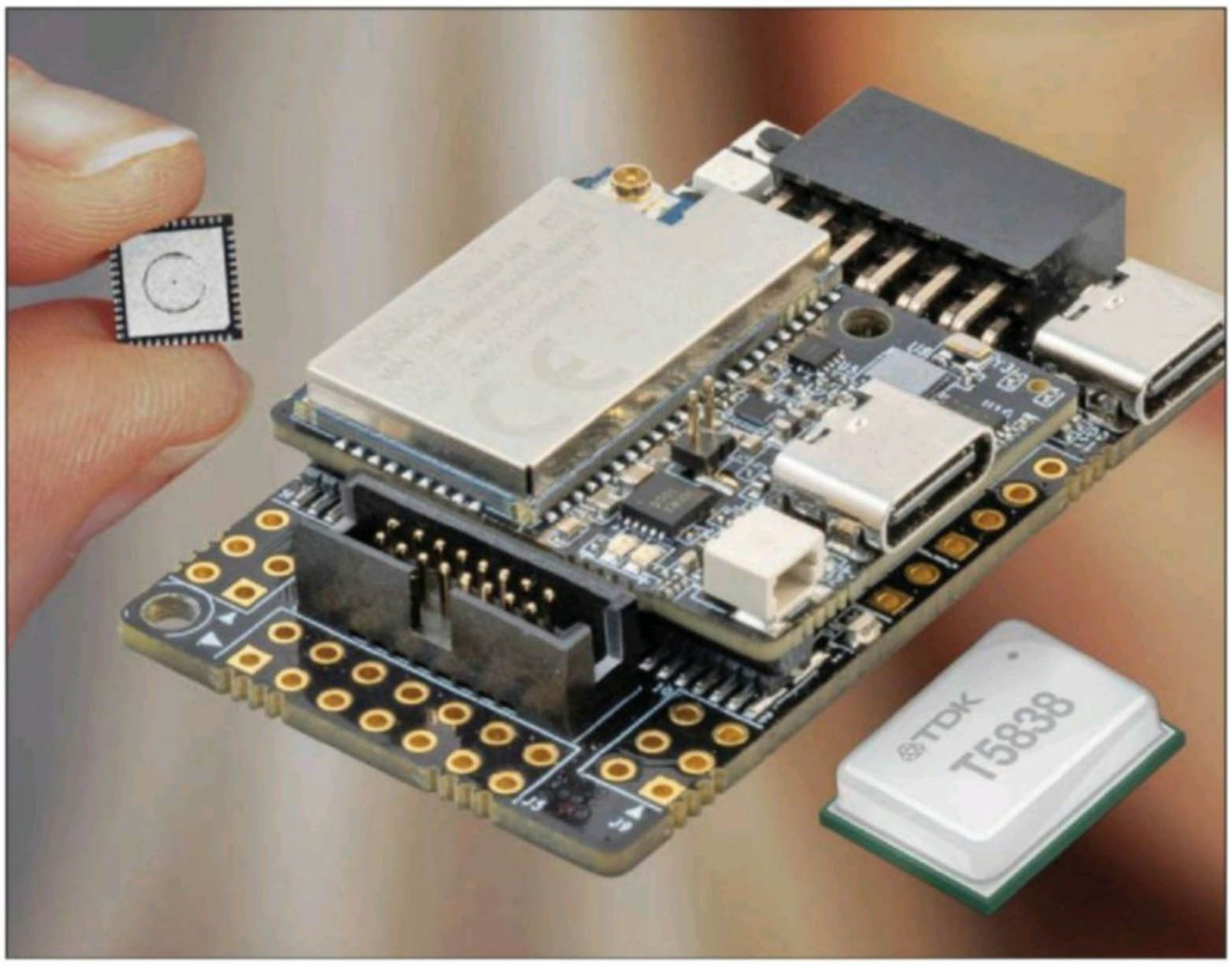


FIGURE 3
The low power consumption and AAD functionalities of the T5838 microphone are essential to our customers who are deploying energy-efficient, high-performance edge AI solutions in speech recognition and acoustic event detection.

additional processing to filter microphone output. This is particularly valuable for devices that require always-on AI systems, such as voice assistants, where power efficiency and processing capability are key.

The SmartSound One, a USB plug-and-play microphone interface module shown in **Figure 2**, is compatible with PDM digital, I²S digital, single-ended analog, and differential analog TDK microphones supplied on a flexible PCB.

MEMS MICROPHONE INNOVATIONS

The consumer electronics industry is seeing rapid advancements in MEMS microphone technology, with manufacturers introducing innovative solutions that push the boundaries of what's possible. Leading consumer electronics companies are leveraging high performance MEMS microphones that deliver superior sound quality and improved power efficiency to enable a wide range of AI-driven features to their customers.

For instance, TDK InvenSense has been at the forefront of MEMS microphone innovation, offering products like the like the T5838 (**Figure 3**) and T5848 [1] that combine high SNR and AOP with ultra-low power consumption. These microphones are equipped with Acoustic Activity Detection (AAD), which enables integration of advanced AI processors while lowering system level power consumption. TDK's MEMS microphones are particularly well-suited for edge AI systems, where power efficiency and performance are paramount.

With these innovations, MEMS microphones enable new use cases across industries, from real-time voice transcription in business settings to voice-activated smart home systems, allowing users to control their environment with simple voice commands. The flexibility and programmability of MEMS microphones make them an essential component in the next generation of voice-enabled devices.

FUTURE OF VOICE ENABLED AI

As AI advances, MEMS microphones' role in enabling voice-activated systems will only grow. MEMS microphones are at the heart of the voice-driven future, from hands-free control in automobiles to real-time translation in smartphones. Their ability to capture sound accurately, even in challenging environments, combined with their ultra-low power consumption, makes them indispensable for a wide range of applications.

With continued innovation in MEMS technology, we can expect to see even more sophisticated use cases emerge in the coming years. Voice-enabled AI is becoming more intuitive and user-friendly, and MEMS microphones are playing a critical role in making this vision a reality. By providing the high-quality sound capture necessary for accurate voice recognition and AI processing, these microphones are unlocking new possibilities for the next generation of consumer electronics.

Learn more about TDK MEMS microphones at invensense.tdk.com/smartsound

Additional materials from the author are available at:
www.circuitcellar.com/article-materials
References [1] as marked in the article can be found there.

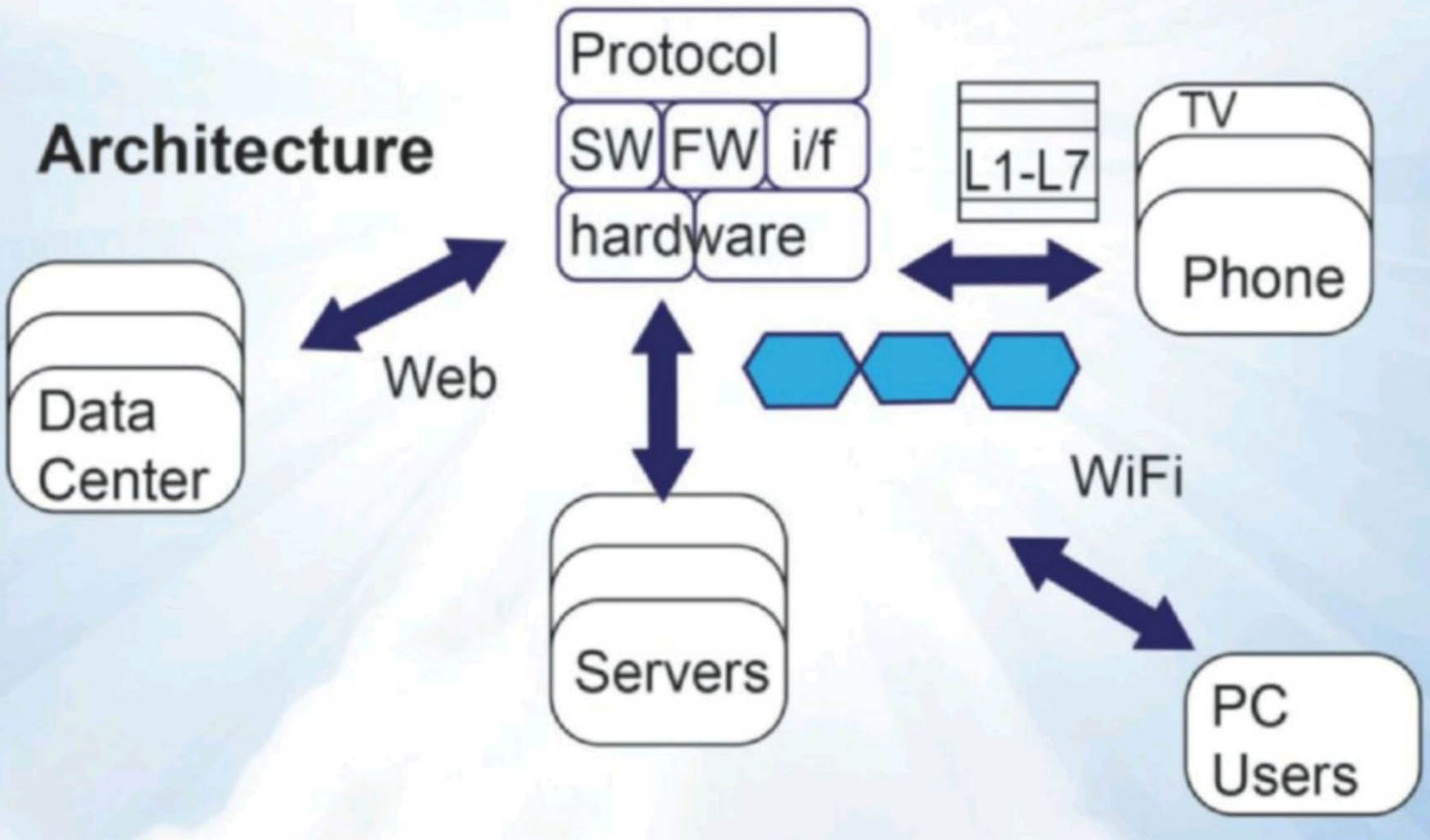
TDK | www.invensense.tdk.com

ABOUT THE AUTHOR

Uday Mudoi is a seasoned technology leader with over 15 years of experience in product management, marketing, and business development. Currently, he serves as Vice President & General Manager of the Microphone Business Unit at TDK InvenSense, where he drives innovation in voice interfaces for AI-enabled devices. With a strong background in both hardware and software development, Uday has a proven track record of success in the semiconductor and IP development industries. He holds a Bachelor of Science in Electrical Engineering from the Indian Institute of Technology, Kharagpur, and an MBA from Columbia University. Uday is passionate about leveraging technology to create impactful solutions and is eager to connect with like-minded professionals.

SeaFire® Micros, Inc.

IP & Technology for Cloud Networks


Architecture
Protocol
SW
FW
i/f
hardware
L1-L7
TV
Phone
Data Center
Web
WiFi
Servers
PC Users

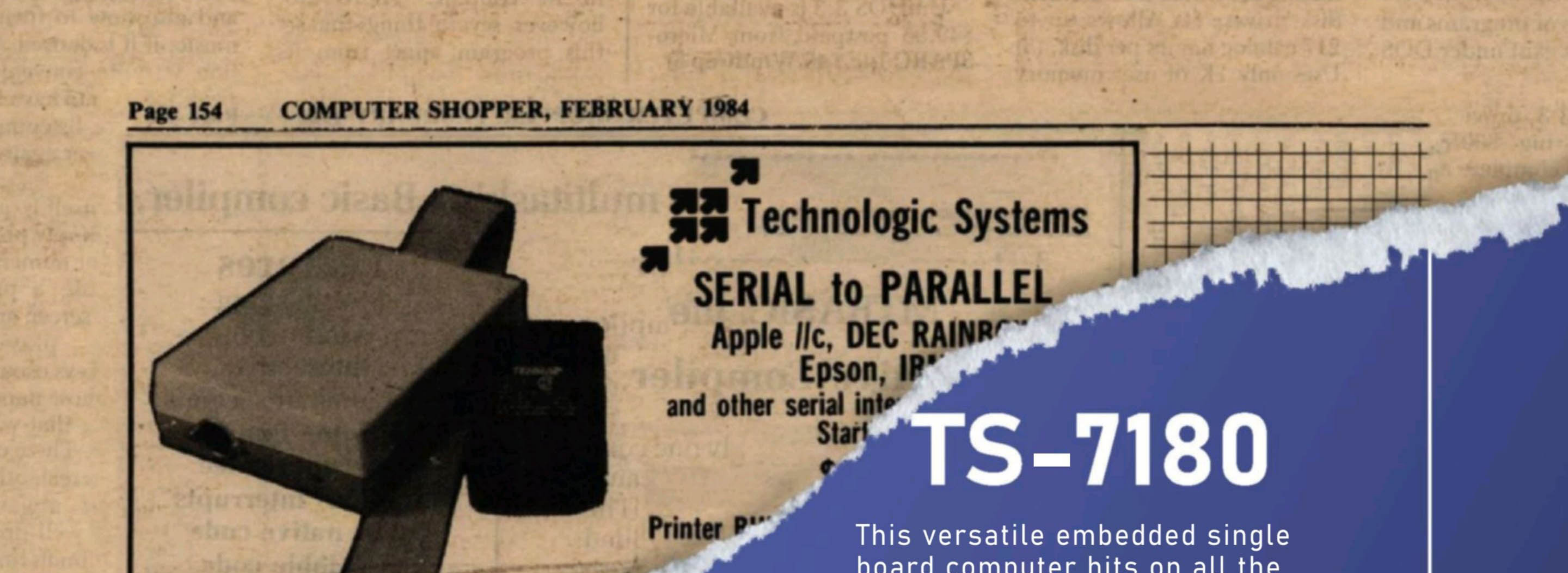
Design through Production
Military — Commercial

Hardware (Analog/Digital)
Firmware • Software • FPGA

Patented Network Technology | Unique Cloud Technology
No Compete Government Procurement
No Compete Government Funding

20%
DISCOUNT
on 1st
Purchase

info@seafire.com | seafire.com



Page 154 COMPUTER SHOPPER, FEBRUARY 1984

Technologic Systems

SERIAL to PARALLEL

Apple //c, DEC RAINB

Epson, IB

and other serial inte

Start

Printer B

Rev.D
TS-7180

TS-7180

This versatile embedded single board computer hits on all the main points for a low power, cost effective, Internet-of-Things (IoT) capable, and ready-to-deploy OEM board with an emphasis on data integrity.

NXP i.MX6UL SOC with a 696 MHz Arm® Cortex®-A7 core

512 MB DDR3 RAM (1 GB DDR3 RAM Option Available)

4 GB MLC eMMC Flash, configurable as pSLC for added reliability

Connected with WiFi, Bluetooth, MultiTech and XBee/NimbeLink

Easily Track Assets with GPS Radio and Cellular Modem

Celebrating 40 Years

From humble beginnings in a garage in Ohio, EmbeddedTS has grown to become a leader in rugged embedded computing solutions. Since 1984, we've proudly remained family-owned and dedicated to delivering high-quality, reliable products.

Every board we sell is still designed, shipped, and supported from right here in the United States, ensuring you get top-tier craftsmanship.

Made in the USA
with global parts

embeddedTS
embeddedTS.com